【第三方模型使用条款】

1.请务必仔细阅读模型包含的使用说明书中记载的使用规则，遵守模型与动作的使用规范。
2.请勿私自修改，二次上传转发，违反道德使用模型，也请遵守制作者标记规定。
3.同人和二次创作必须遵守版权法律规范。
4. 若用户因违反本应用程序使用条款，或违反模型使用规则，而造成侵犯版权的情况。用户将自行承担所有法律责任，我们不承担任何责任。

【VRM模型汇入步骤】

1.将你的.vrm放入以下路径即可
(你的路径)\wakaru\Models\VRM\(你的.vrm)

2. 汇入后表情按名单顺序自动对应捕捉
如果缺漏则不自动对应捕捉，但不会有任何错误
VRM为对应Preset

- 眼:["Blink"]
- 唇:["A","I","U","E","O"]

因为VRM没有对表情做分类，因此新增表情UI可用滑鼠拖拉来变更分类的功能

Wakaru steam默认路径

64-bit: C:\Program Files (x86)\Steam\steamapps\common\wakaru

32-bit: C:\Program Files\Steam\steamapps\common\wakaru

【VRM模型制作与修改教程】

VRM是日本近年在VR，Vtuber领域大力推荐的3D模型格式，2018年3月左右公开，还非常新颖，日文区以外的使用者还非常少，日文以外的教程，研究在网路上也不多。
既然我开发的wakaru采用了这个格式，因此有义务写一些介绍与教程，这个系列只会包含粗浅，对于3D模型略有知识的人都能够看懂的程度，太过深入的进阶内容，还请多多看官网。

本篇部分引用，翻译自官方网站
官方:
https://dwango.github.io/

简介:

因为人形3D模型格式多种多样的不统一，坐标系不同，尺度不同，初始姿势不同，表情不同，骨骼导入程序的方法也不同。对于VR开发者来说模型的处理非常麻烦，虽然FBX可以被大多数3D软件输出与读取，但是缺乏许多对于VR应用以及游戏应用的必要资讯，VRM转档程式会自动补足这些缺少的部分，使人形3D模型的应用更加简单。

架构:

VRM基于3D标准格式glTF2.0，这是包含处理人形模型的规范和扩展的格式。
实际上~.vrm直接更改副档名为.glb可以被win10读取，支援glTF2.0的软体应该也可以，不过一些拓展的资讯可能不会被读取。

主要搭载的模型资讯有
纹理Texture
材质Material
着色器Shaders
网格Mesh (Vertex array、index array)
变形，表情 Blend shapes
蒙皮Skinning (4weight)
骨头结构 Node 、bones、rig
第一人称资讯 First-person setting
碰撞与弹簧物理骨骼Secondary (Spring bone、Collision detection)
授权与作者情报 Metadata

应用方法:

VRM目前官方只提供Unity的SDK用于读取和写入制作VRM，但VRM本身与平台无关。可以在其他系统引擎和环境中处理。已有第三方的UE4 SDK正在开发中
官方Unity SDK https://github.com/dwango/UniVRM/releases
UE4 SDK https://github.com/uetokyo/UnrealVRM

制作VRM主要有几个方法
1. 直接从能够输出VRM的软体制作，如VRoid，ブイカツ，目前还很少，导出后可能也免不了进unity修正
2. 由主流3D软体导出成glTF2.0，再用用官方SDK导入Unity，再转换成vrm
3. 由主流3D软体导出成fbx档案，直接导入Unity，再转换成vrm

本教学以fbx作为范例，无论来源，导入Unity后流程都相同
要被导入Unity的原始模型必须已经被设定好绑骨rig以及Blend shape

工作流程:

1.3D软体制作FBX导出，包含blend shapes与Rigged bones
2.FBX导入Unity，第一次转档成VRM，这部分建议以UniVRM v0.35来做，新版本(v0.36~v0.40)会造成转换后脸部错位，后续步骤可以用新版本
3.VRM读入Unity，各项设定(Metadata，Blend shape，Secondary，Shaders...)
4.第二次转档成VRM，完成

本系列预计着重于在模型已经完成的前提下的简易修改的教学

教程:

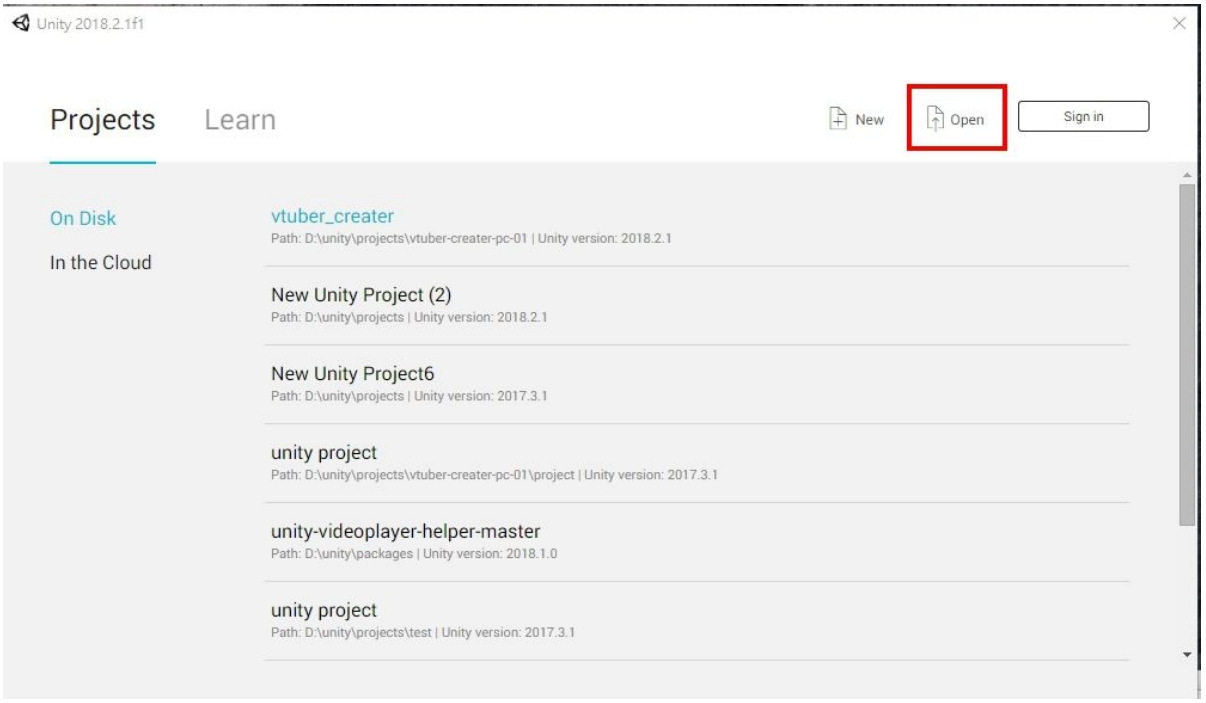
【a. VMR读取与载入与Metadata设定】

那么开始尝试从一个完成的fbx档案，转换成vrm档，这边使用免费的Unity Chan作为范例
如果您单纯只是想要修改现有的vrm档案，可以跳过步骤a-3~a-6

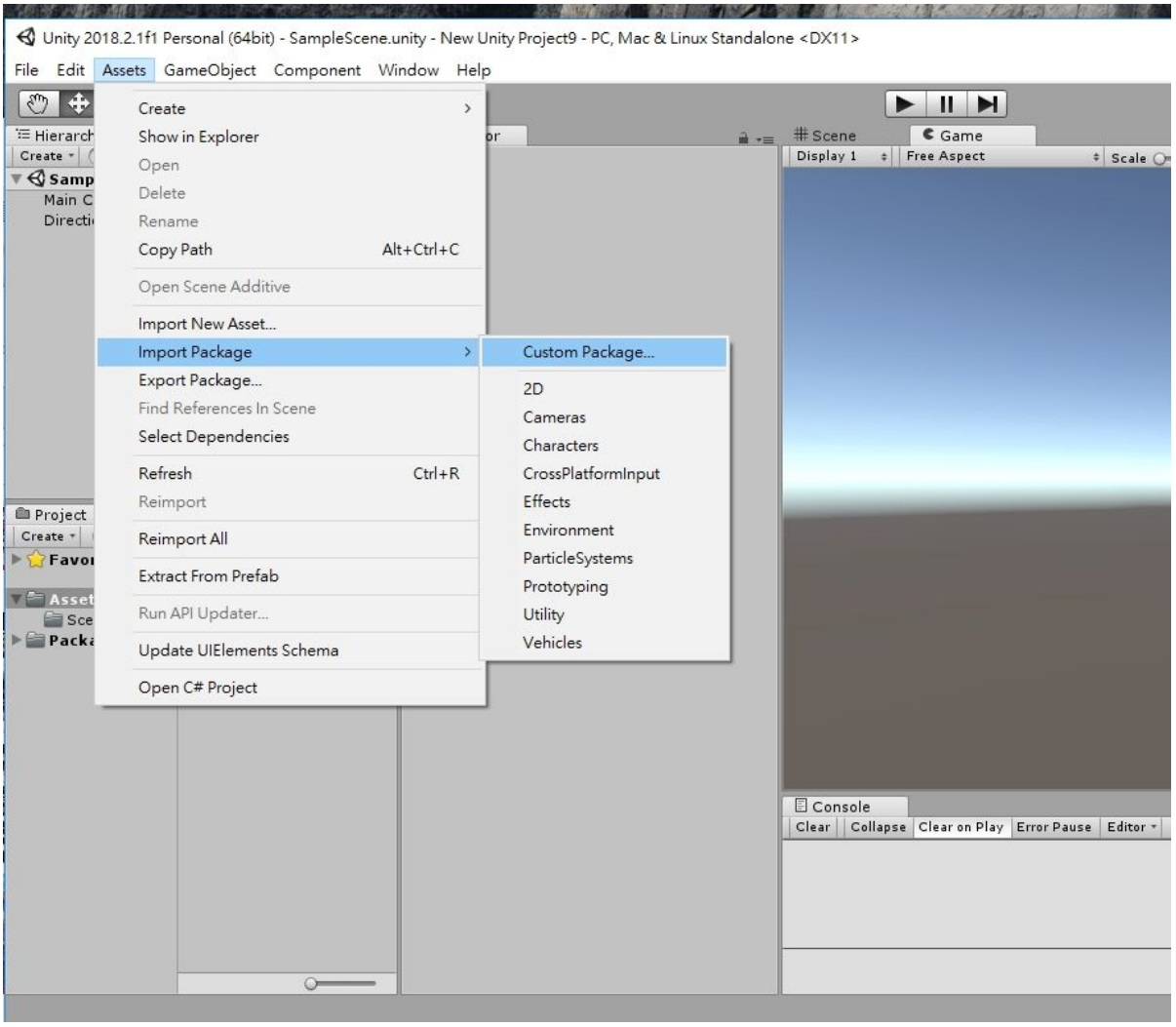
本教程使用版本:

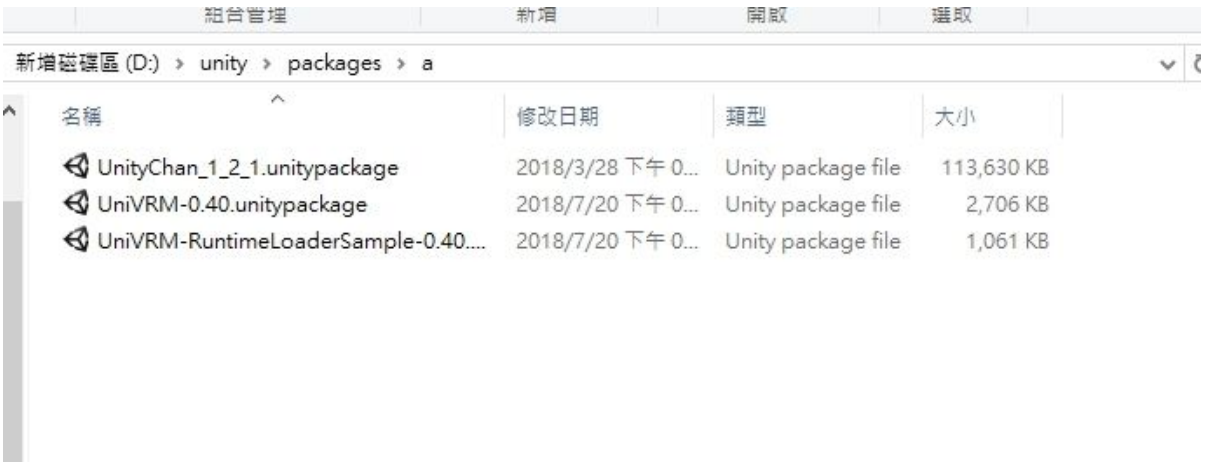
Unity 2018.2.1.f1
本教程会用到插件:
UniVRM v0.40
UniVRM-RuntimeLoaderSample-0.40
UnityChan 1.2.1
请先下载好以上unity package

a-1. Unity 开启新专案， 类型选择3D

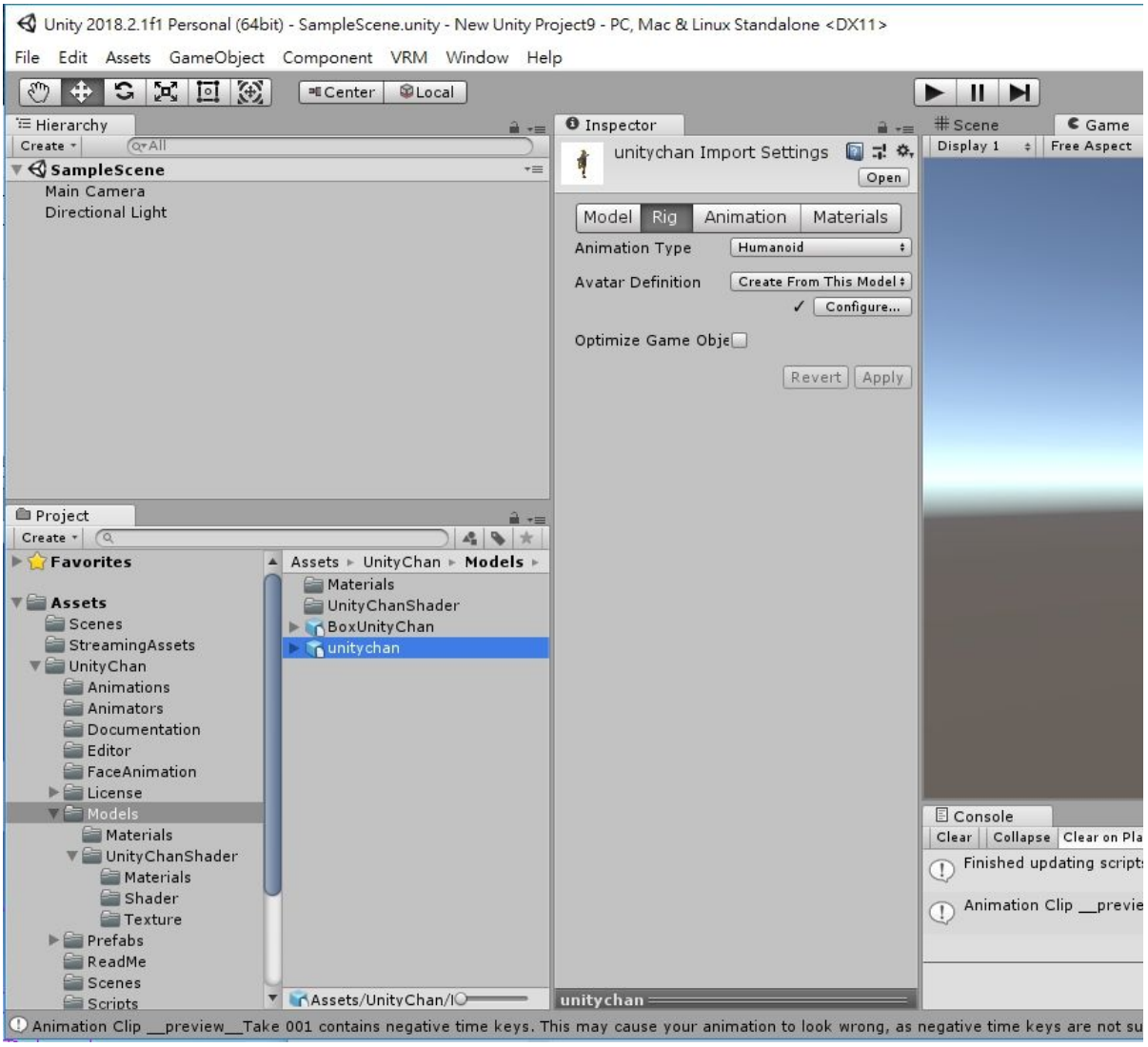


a-2. 载入准备好的3个 unity package，过程如果有对话窗，都直接点选 “import”或”go ahead”

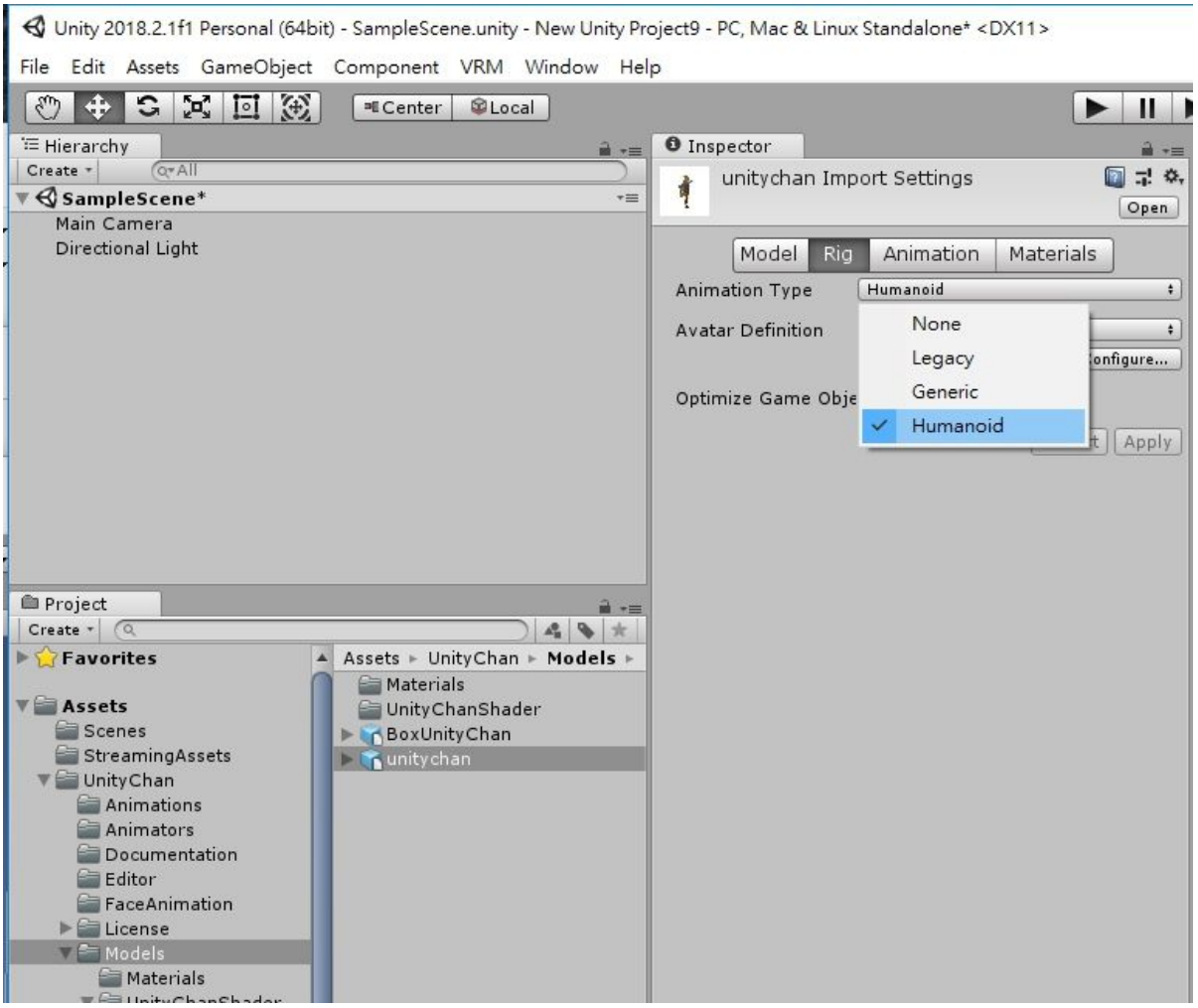




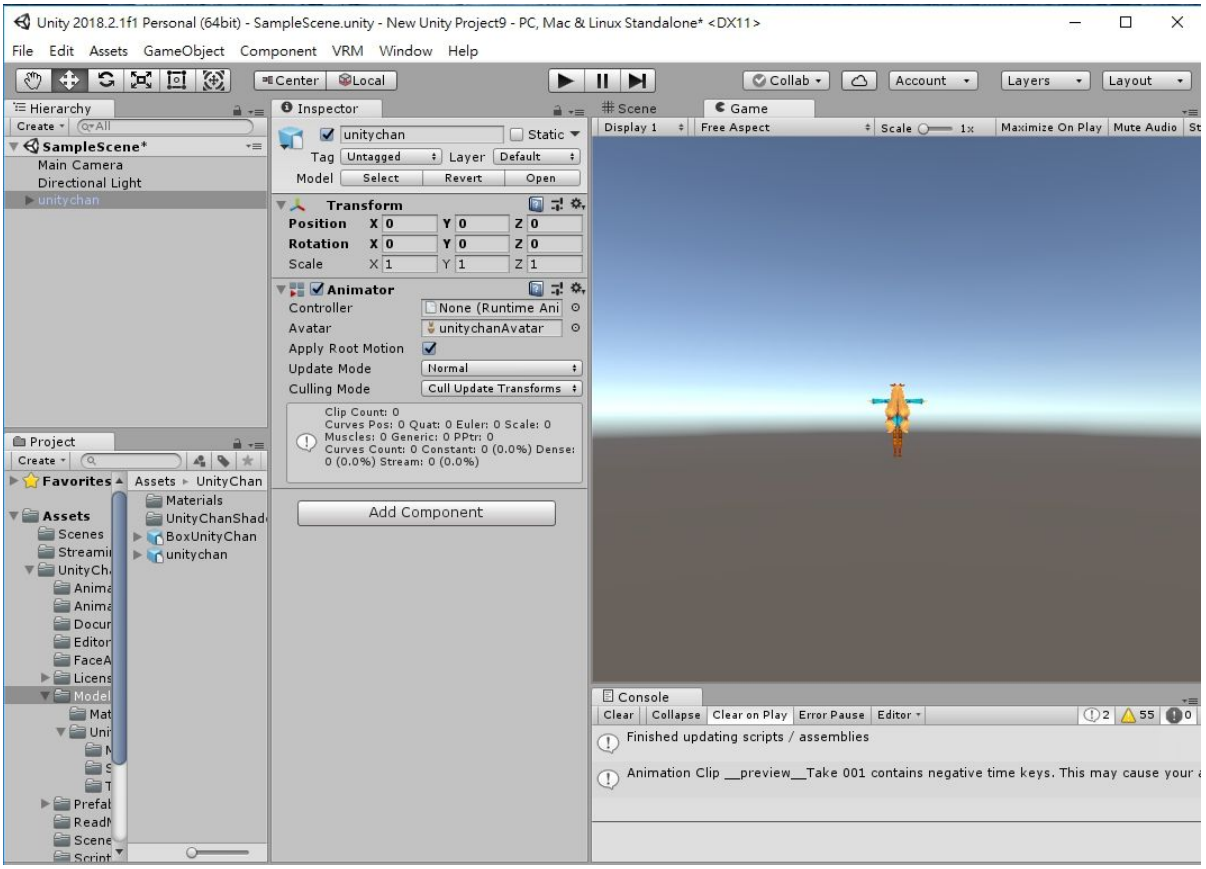
a-3.找到UnityChan的Fbx档案



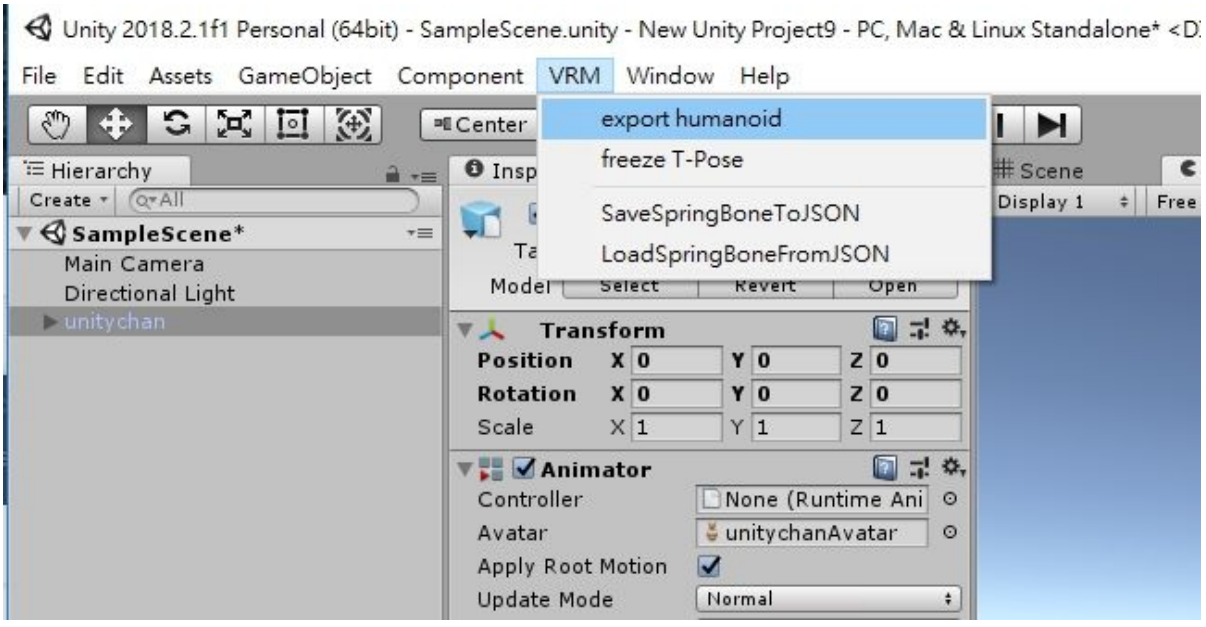
a-4.在Inspector视窗选择Humanoid并点选apply，Unity Chan的已经设定过了，所以可以跳过这一步



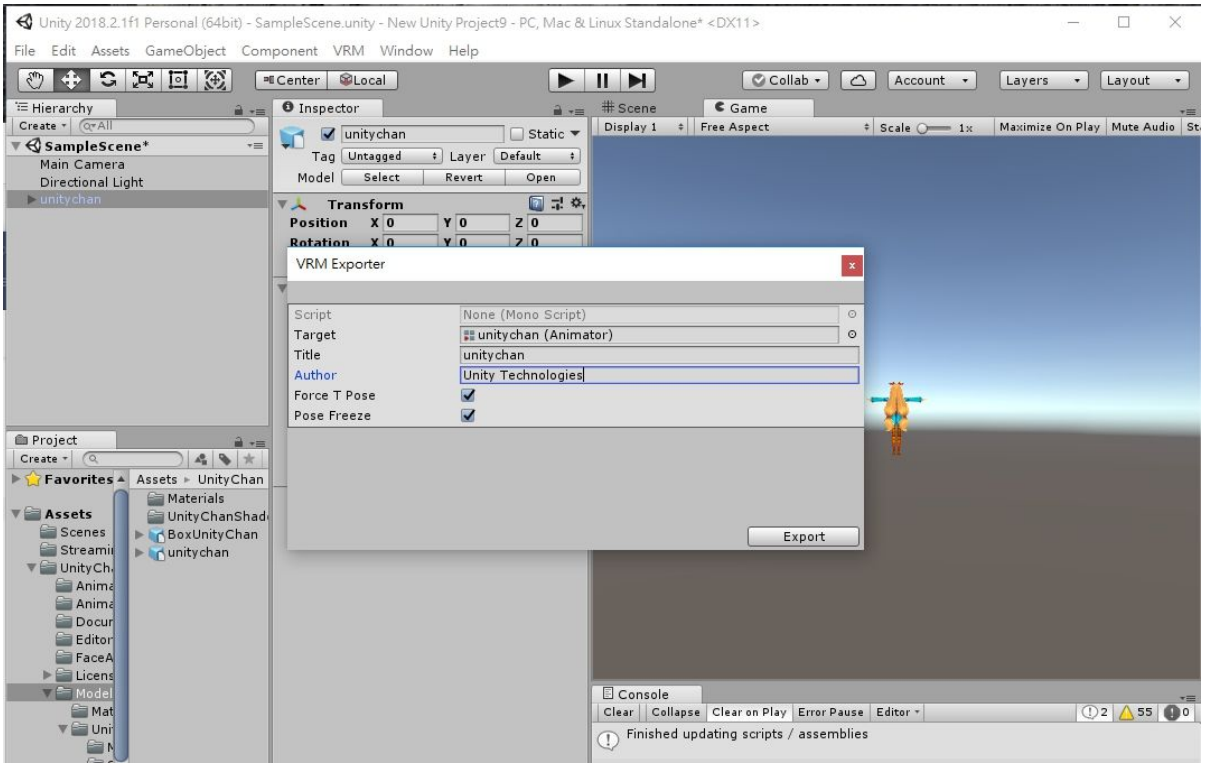
a-5将Unity Chan的fbx拖拉进Hierarchy视窗



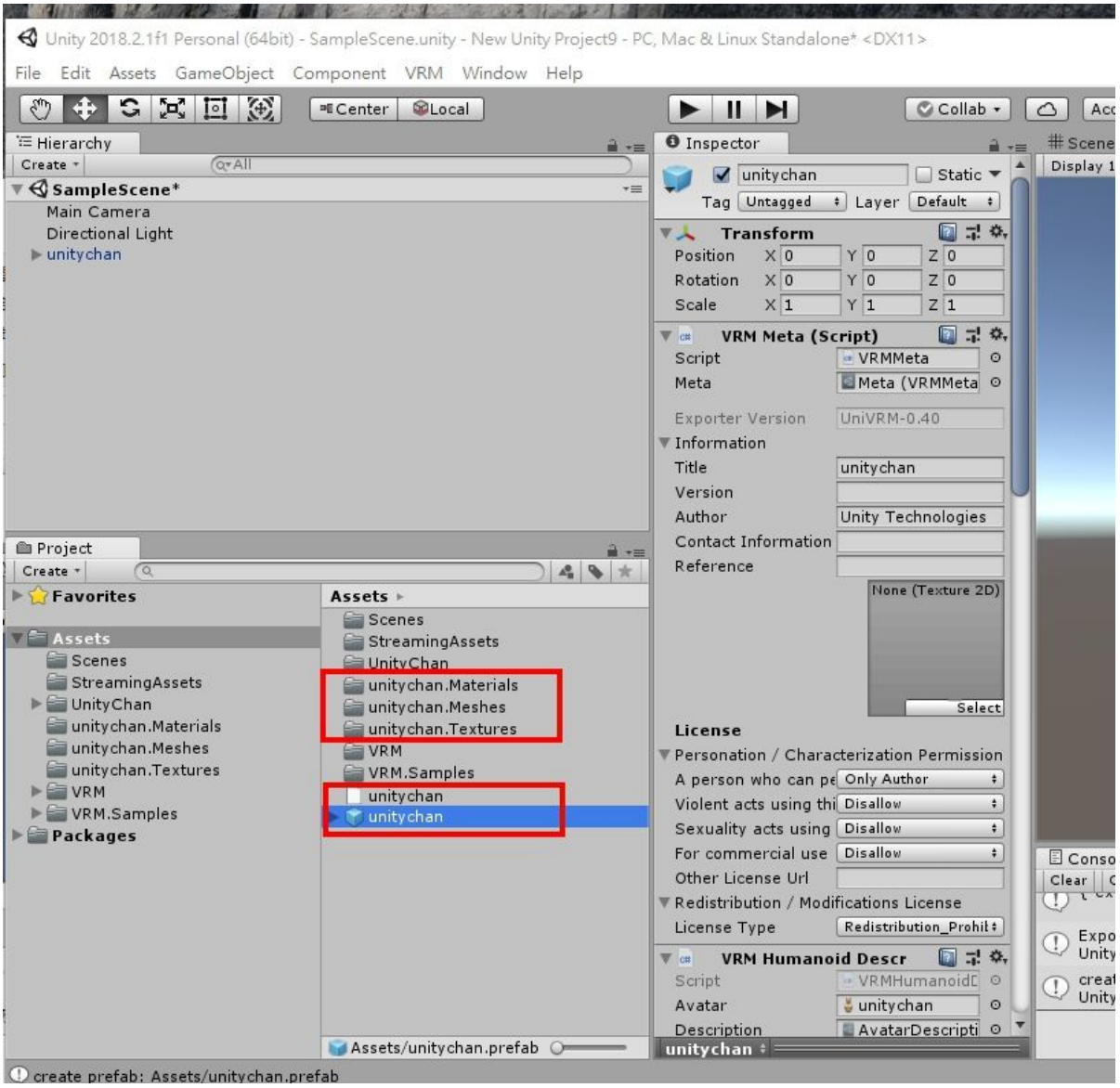
a-6 输出成第一次的VRM



必须要有作者才能输出，这边我就写上Unity Chan的原作者，记得输出在Assets内，才能被Unity编辑



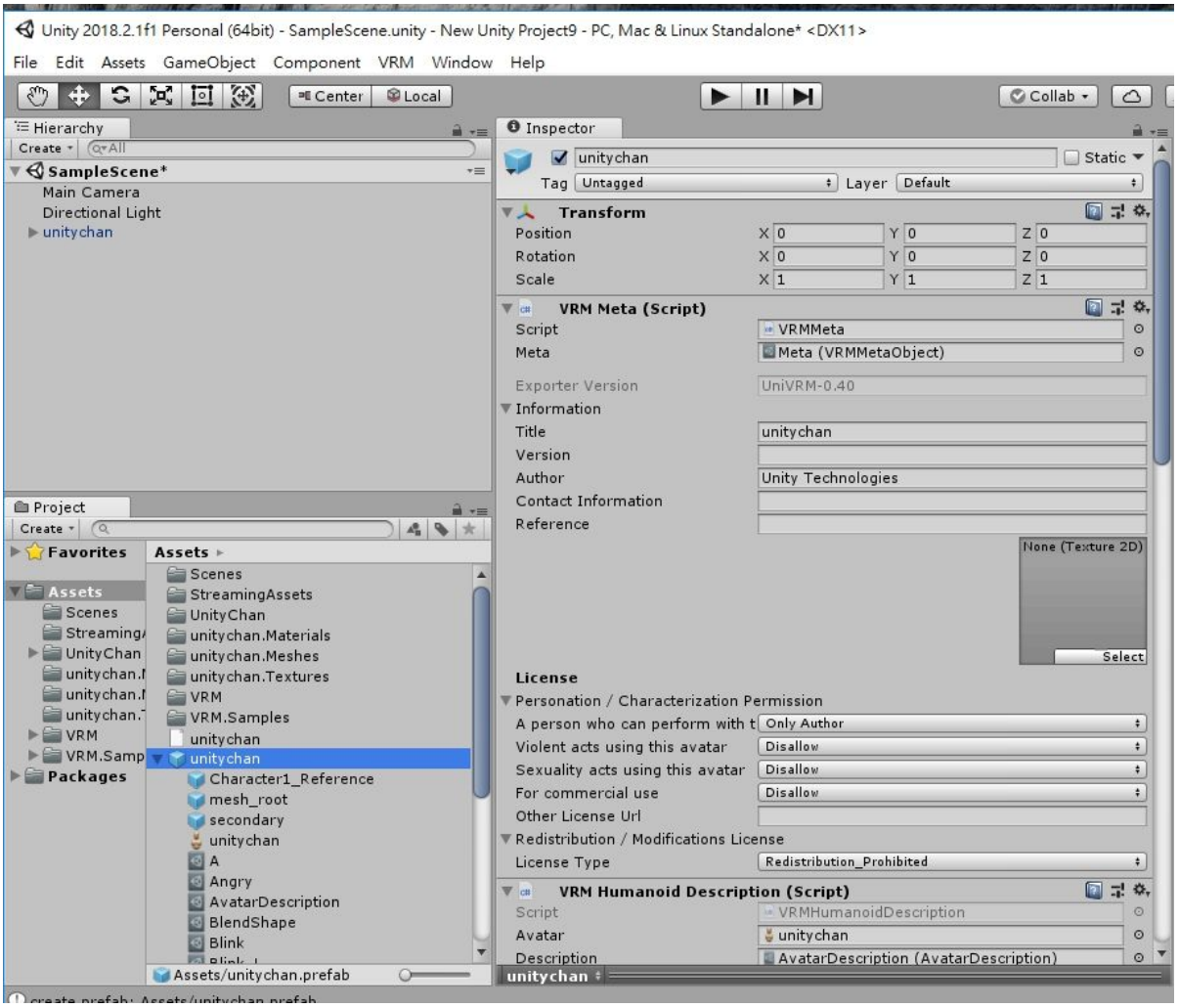
a-7.在Assets资料夹内，只要读取到VRM档案就会自动产生这些新的档案
其中最重要的是这个~.prefab档案以及~.Materials



如果~.prefab拖拉进Hierarchy视窗就会成为一个gameObject
部分对gameObject会提示中断与~.prefab的连结，中断连结之后如果要输出成VRM
记得选择gameObject为对象，而不是~.prefab

Blend shape设定只能在~.prefab，因为预设的unity 编辑器无法编辑~.prefab内层
Secondary设定只能在gameObject
Metadata设定则不限
Shaders则可能都需要
因此各项设定之间可能需要多次再输出覆盖VRM作为存档

a-8 设定metadata
看到Inspector视窗内的VRM Meta (Script)



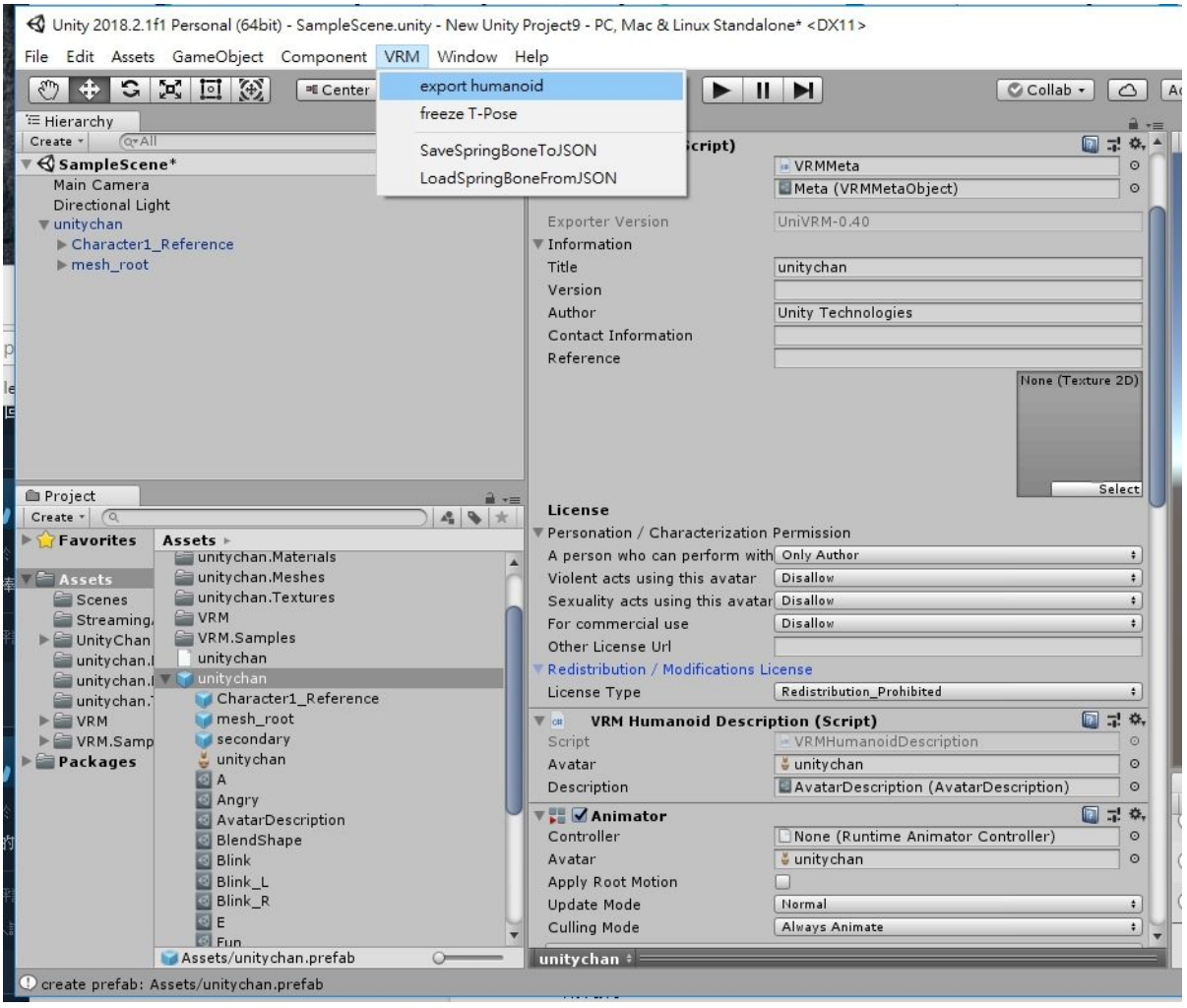
Information这边都是直接填写文字，没什么特别的
方框可以设定预览缩图

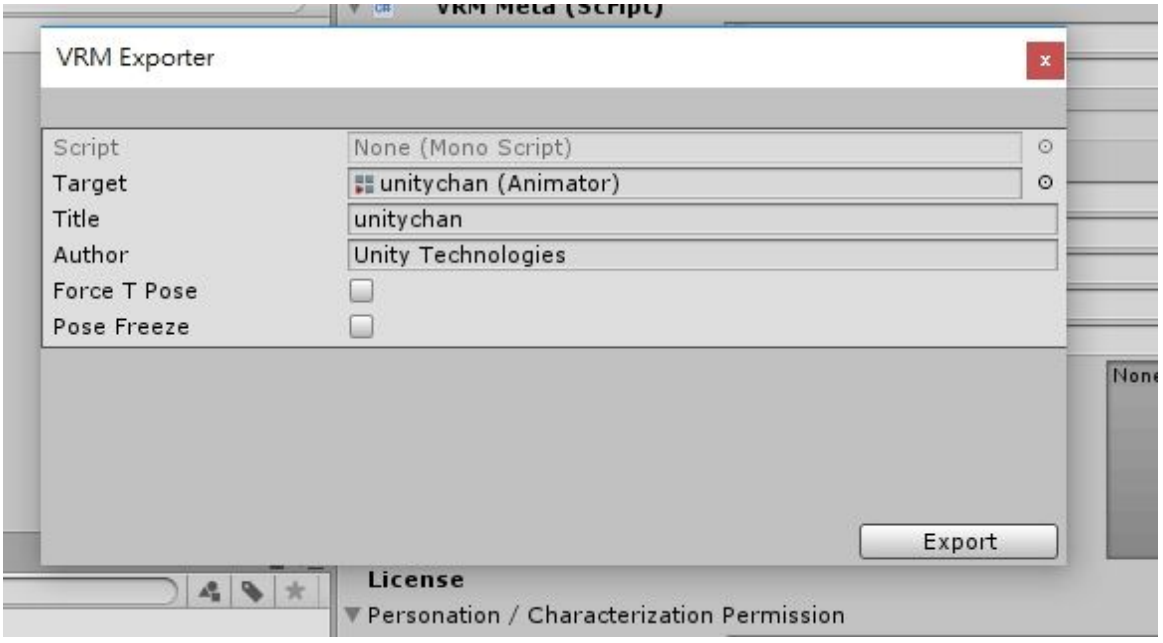
Licence这边几个比较重要的
Personation / Characterization Permission 人格/表征许可
A person who can perform with this avatar能够使用此头像进行表演的人

- Only Author只有作者
- Explicitly Licensed Person特定许可人
- Everyone所有人

Permission to perform violent acts with this avatar允许使用于暴力内容
Permission to perform sexual acts with this avatar允许使用于色情内容
For commercial use允许使用于商业用途
Redistribution Prohibited 可否公开上传与改造

a-9 设定完成后，如果不继续进行其他编辑，选择此~.prefab，第二次输出成VRM，可以直接覆盖第一次的


Unity 2018.2.1f1 Personal (64bit) - SampleScene.unity - New Unity Project9 - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component VRM Window Help
export humanoid
freeze T-Pose
SaveSpringBoneToJSON
LoadSpringBoneFromJSON
Hierarchy
SampleScene*
Main Camera
Directional Light
unitychan
Character1_Reference
mesh_root
Project
Favorites
Assets
Scenes
UnityChan
VRM
VRM.Samples
unitychan
Packages
Character1_Reference
mesh_root
secondary
A
Angry
AvatarDescription
BlendShape
Blink
Blink_L
Blink_R
E
Fun
Assets/unitychan.prefab
create prefab: Assets/unitychan.prefab
VRMMeta
Meta (VRMMetaObject)
Exporter Version
UniVRM-0.40
Information
Title
unitychan
Version
Author
Unity Technologies
Contact Information
Reference
None (Texture 2D)
Select
License
Personation / Characterization Permission
A person who can perform with
Only Author
Violent acts using this avatar
Disallow
Sexuality acts using this avatar
Disallow
For commercial use
Disallow
Other License Url
Redistribution / Modifications License
License Type
Redistribution_Prohibited
VRM Humanoid Description (Script)
Script
VRMHumanoidDescription
Avatar
unitychan
Description
AvatarDescription (AvatarDescription)
Animator
Controller
None (Runtime Animator Controller)
Avatar
unitychan
Apply Root Motion
Update Mode
Normal
Culling Mode
Always Animate


VRM Exporter
Script
None (Mono Script)
Target
unitychan (Animator)
Title
unitychan
Author
Unity Technologies
Force T Pose
Pose Freeze
Export
License
Personation / Characterization Permission

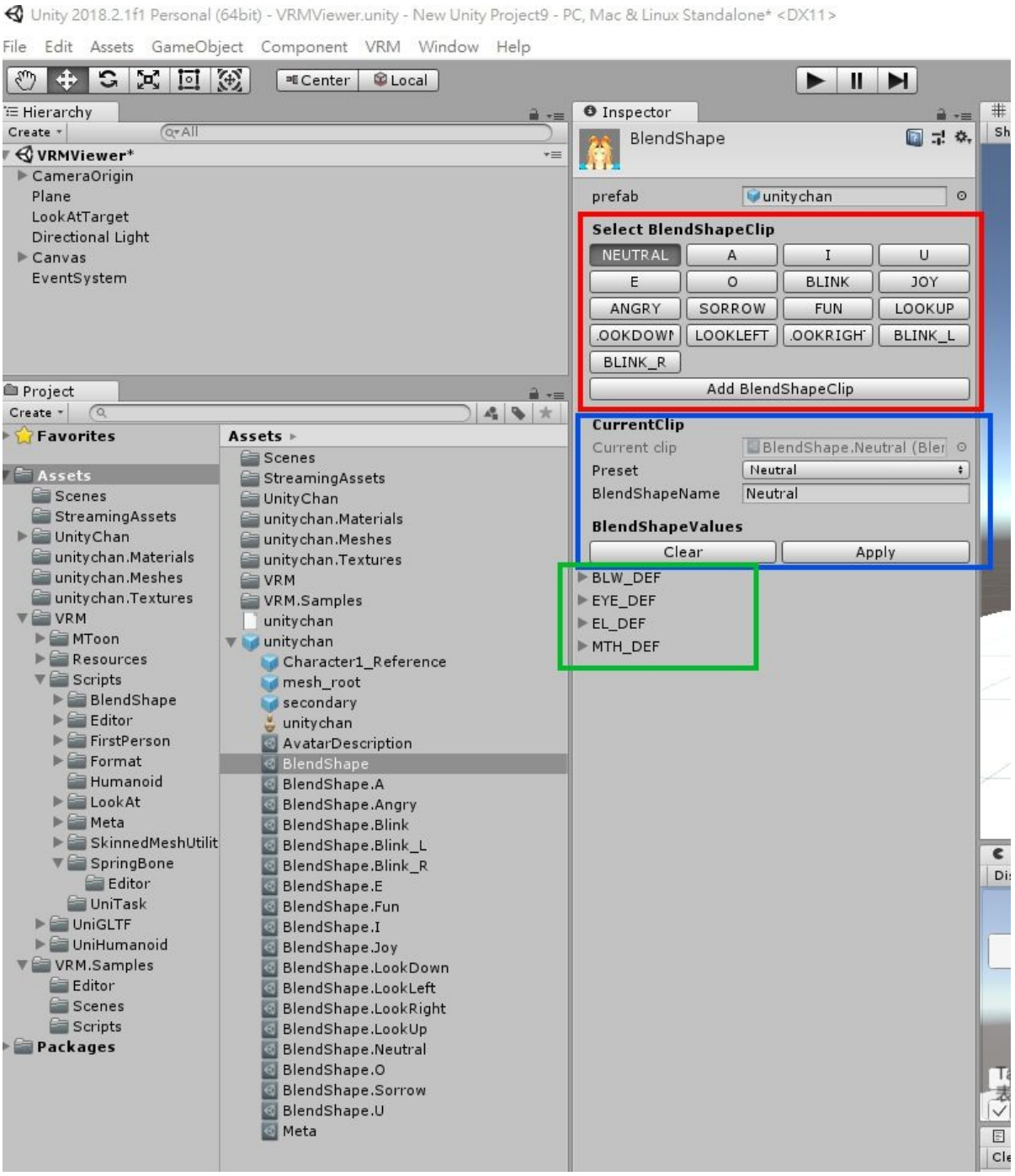
官方文档

https://dwango.github.io/en/vrm/how_to_make_vrm/#how-to-create-vrm-file-conversion-from-existing-3d-model

【b. Blend shape设定】

Blend shape设定只能在~.prefab，所有表情设定都在此处

b-1. 展开 untiychan的prefab， 点选内部的BlendShape，在Inspector可以看到编辑介面



红色为所有VRM的预设表情，目前每一个还是空的
蓝色为正在被选择编辑中的表情
绿色为来自于原始模型做好的Blend shape或Morph，VRM的Blend shape实际上为一个统一的对外介面，来包装这些原始的Blend shape，由这些原始Blend shape组合成每一个Preset

VRM的预设表情各自属于与预设名称相同的Preset，这些Preset因为统一规格，通常会被优先使用

全脸的Preset:
- NEUTRAL 待机表情
- Joy, Angry, Sorrow, Fun 喜怒哀乐

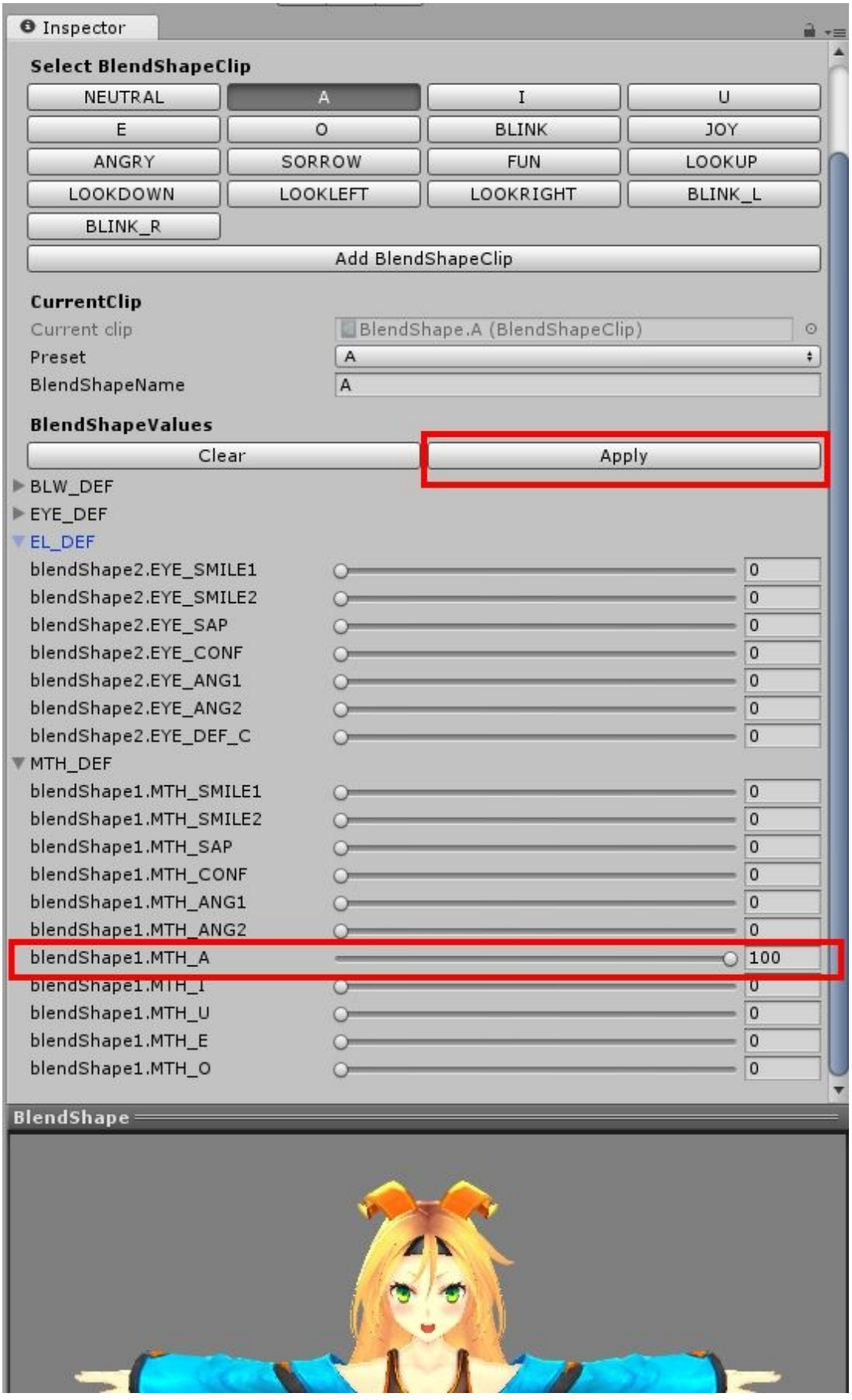
眼的Preset:
- Blink, 眨双眼
- Blink_L, Blink_R 眨单眼

口的Preset:
- A, I, U, E, O

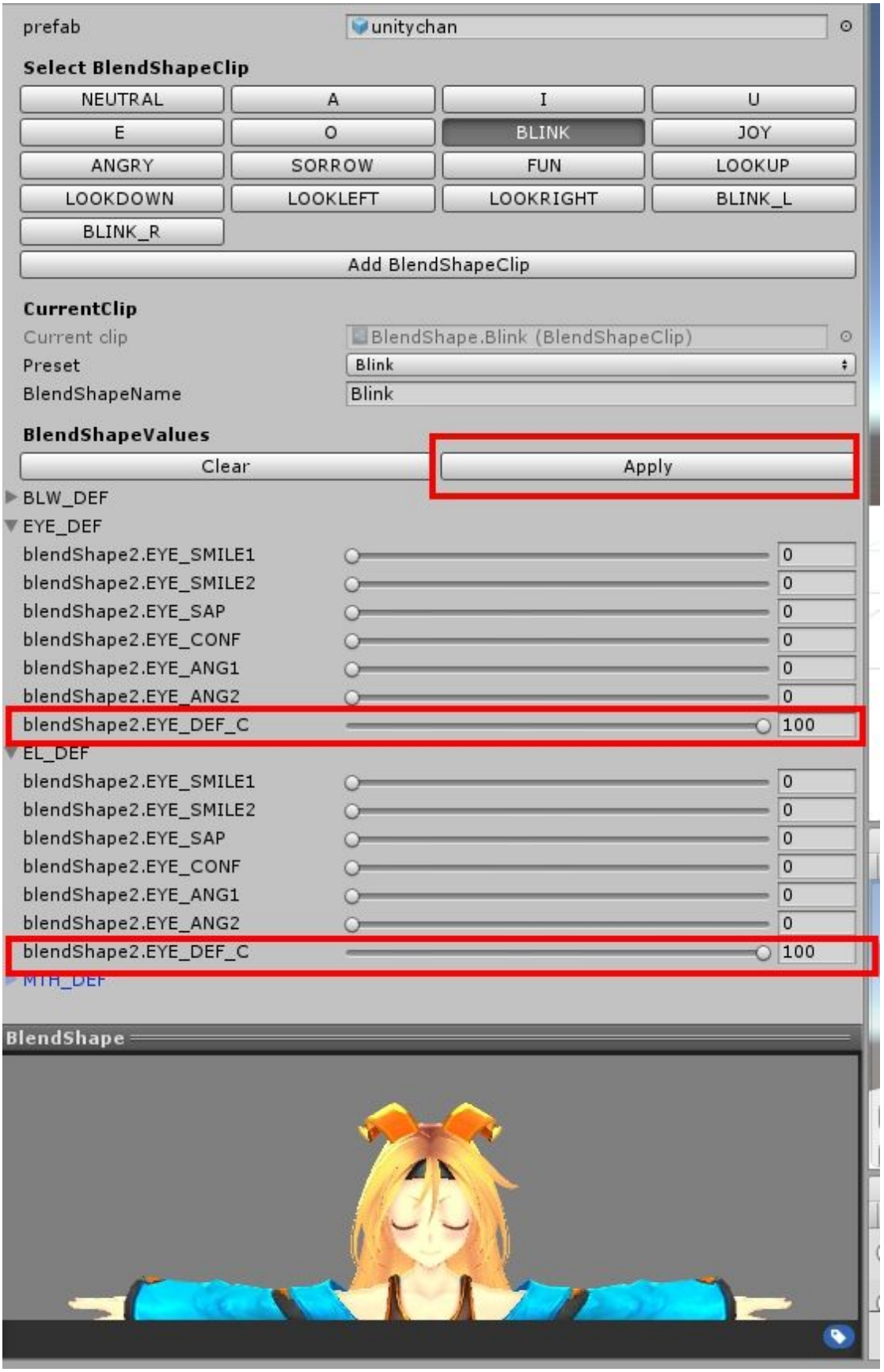
视线的Preset:
- LookUp, LookDown, LookLeft, LookRight 如果视线由Blend shape控制时使用

其他新创造的Preset:
- Unknown 其他新创造的Blend shape都属于这个

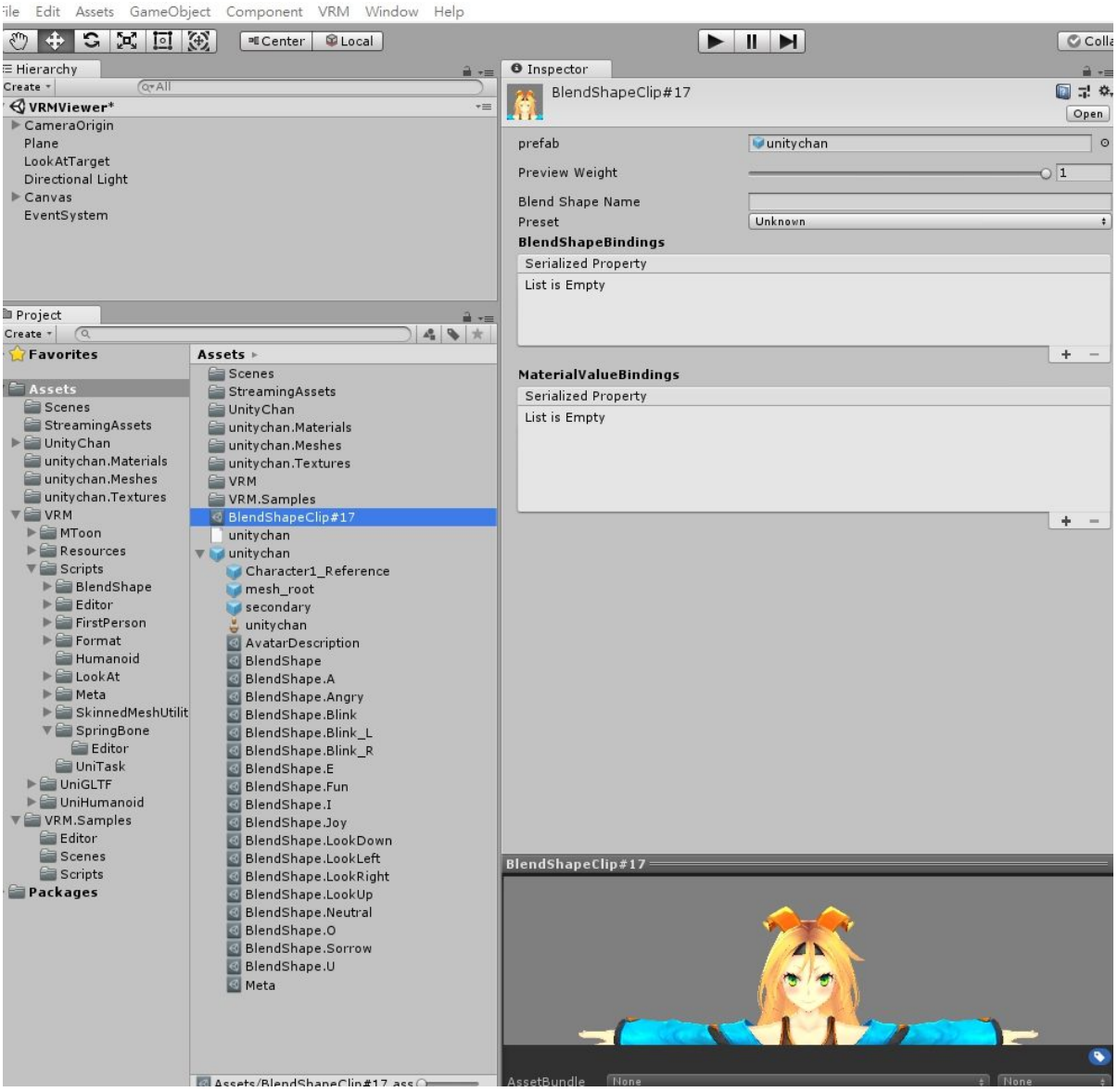
b-2.现在每一个表情都是空的，设定其中几个作为范例
Blend shape A，设好原始Blend shape的数值后Apply，就完成了


Inspector
Select BlendShapeClip
NEUTRAL
A
I
U
E
O
BLINK
JOY
ANGRY
SORROW
FUN
LOOKUP
LOOKDOWN
LOOKLEFT
LOOKRIGHT
BLINK_L
BLINK_R
Add BlendShapeClip
CurrentClip
Current clip
BlendShape.A (BlendShapeClip)
Preset
A
BlendShapeName
A
BlendShapeValues
Clear
Apply
BLW_DEF
EYE_DEF
EL_DEF
blendShape2.EYE_SMILE1
blendShape2.EYE_SMILE2
blendShape2.EYE_SAP
blendShape2.EYE_CONF
blendShape2.EYE_ANG1
blendShape2.EYE_ANG2
blendShape2.EYE_DEF_C
MTH_DEF
blendShape1.MTH_SMILE1
blendShape1.MTH_SMILE2
blendShape1.MTH_SAP
blendShape1.MTH_CONF
blendShape1.MTH_ANG1
blendShape1.MTH_ANG2
blendShape1.MTH_A
100
blendShape1.MTH_I
blendShape1.MTH_U
blendShape1.MTH_E
blendShape1.MTH_O
BlendShape

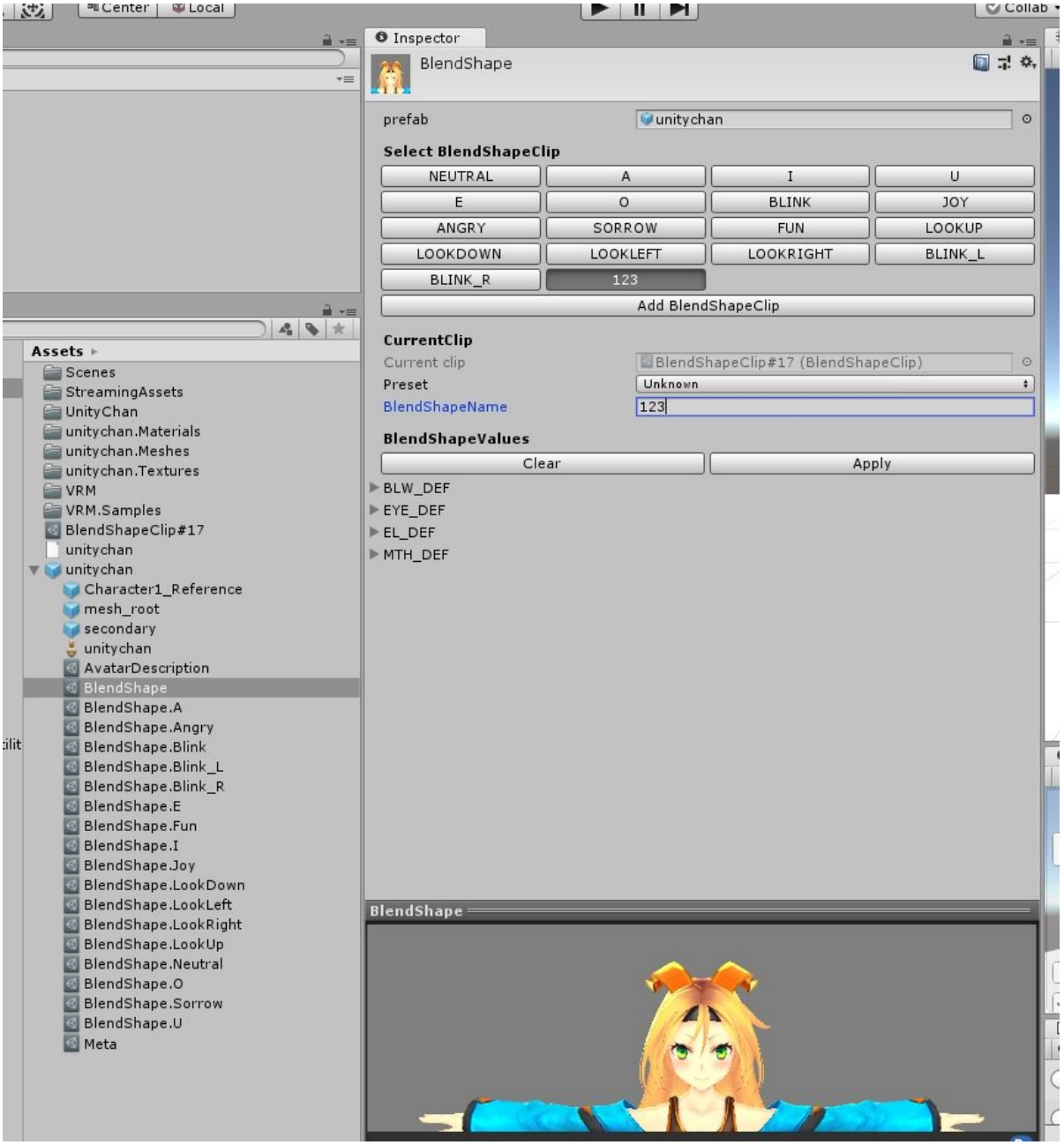
Blend shape Blink


prefab
unitychan
Select BlendShapeClip
NEUTRAL
A
I
U
E
O
BLINK
JOY
ANGRY
SORROW
FUN
LOOKUP
LOOKDOWN
LOOKLEFT
LOOKRIGHT
BLINK_L
BLINK_R
Add BlendShapeClip
CurrentClip
Current clip
BlendShape.Blink (BlendShapeClip)
Preset
Blink
BlendShapeName
Blink
BlendShapeValues
Clear
Apply
BLW_DEF
EYE_DEF
blendShape2.EYE_SMILE1 0
blendShape2.EYE_SMILE2 0
blendShape2.EYE_SAP 0
blendShape2.EYE_CONF 0
blendShape2.EYE_ANG1 0
blendShape2.EYE_ANG2 0
blendShape2.EYE_DEF_C 100
EL_DEF
blendShape2.EYE_SMILE1 0
blendShape2.EYE_SMILE2 0
blendShape2.EYE_SAP 0
blendShape2.EYE_CONF 0
blendShape2.EYE_ANG1 0
blendShape2.EYE_ANG2 0
blendShape2.EYE_DEF_C 100
MTH_DEF
BlendShape

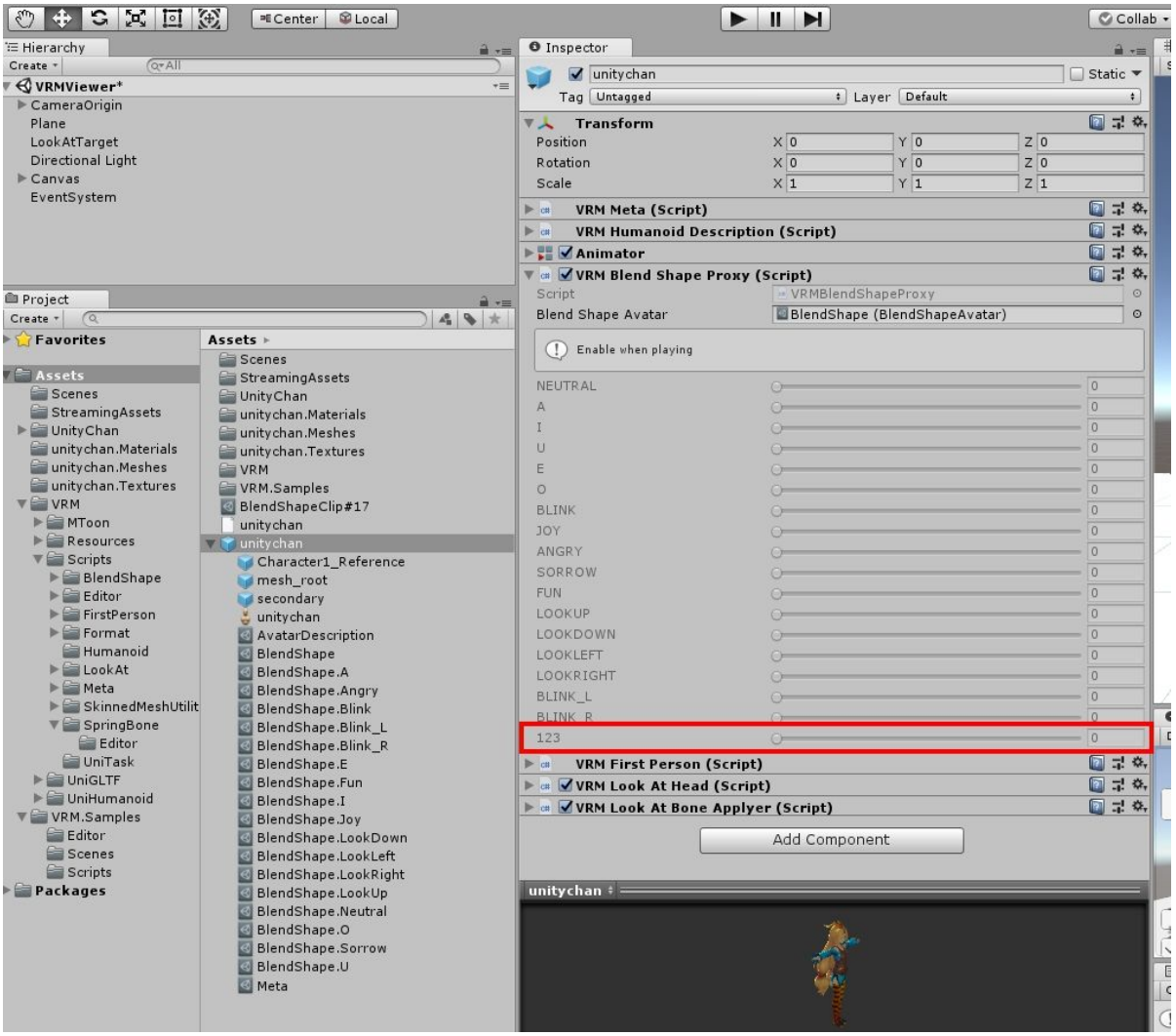
b-3 建立新表情 点选 Add BlendShapeClip，建立一个BlendShapeClip档案
这是刚建立的



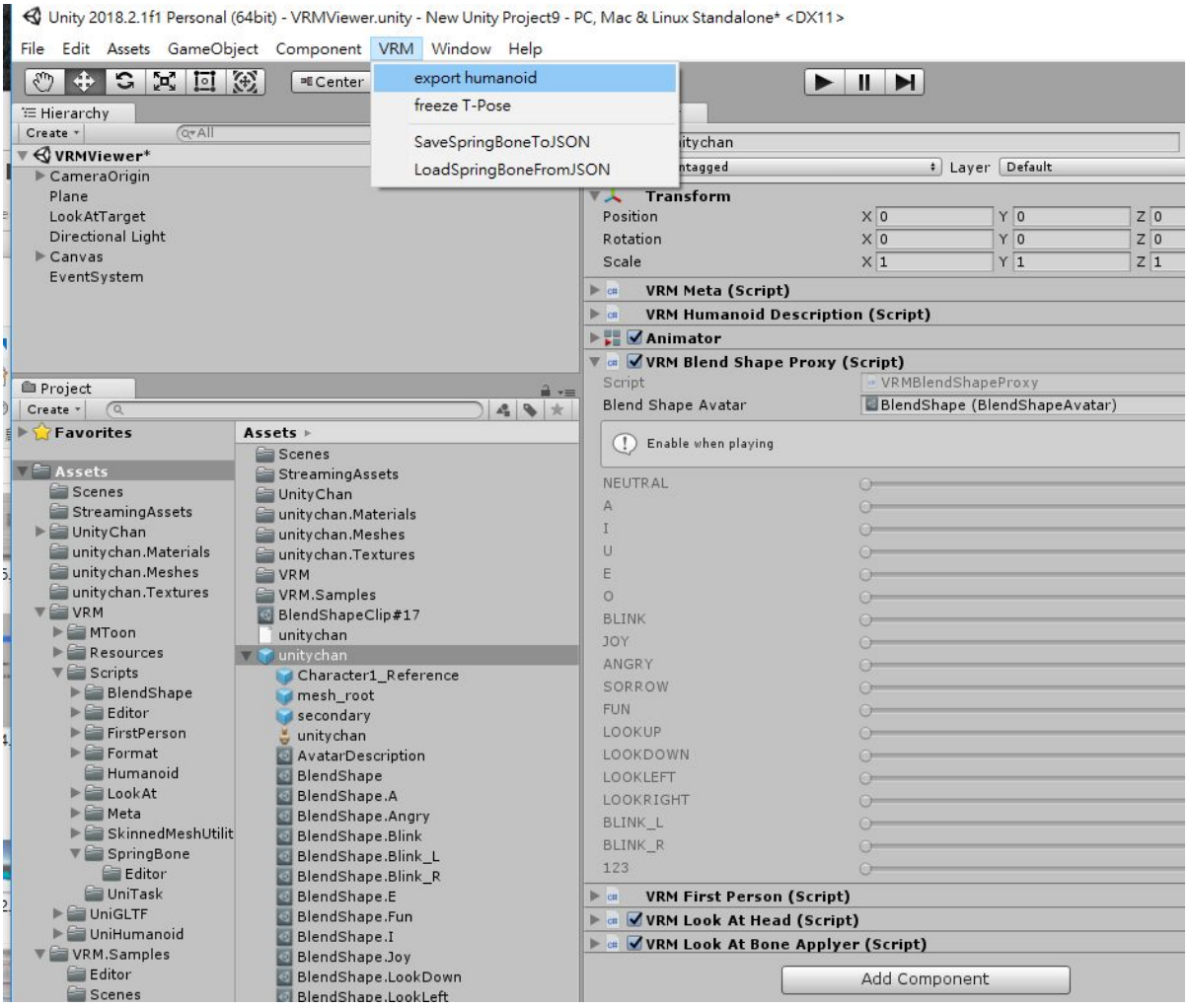
b-4 回到Blend shape 介面，会看到一个空的Blend shape，Preset是Unknow
这时就可以自己命名BlendShapeName，一样设定完后Apply



b-5 回到~.prefab，可以看到新的Blend shape确实新增了，在Play状态就可以使用



b-6. 记得输出VRM作为存档

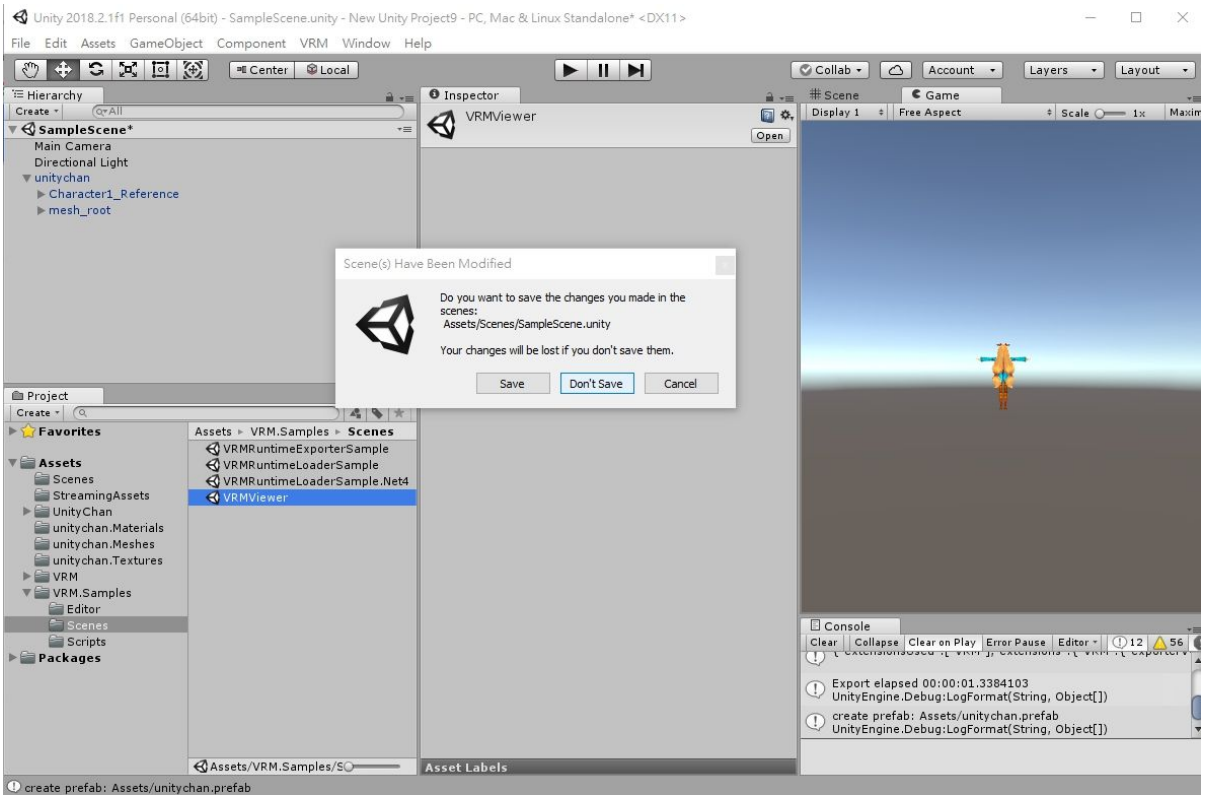


官方文档
https://dwango.github.io/en/vrm/univrm/components/univrm_blendshape/

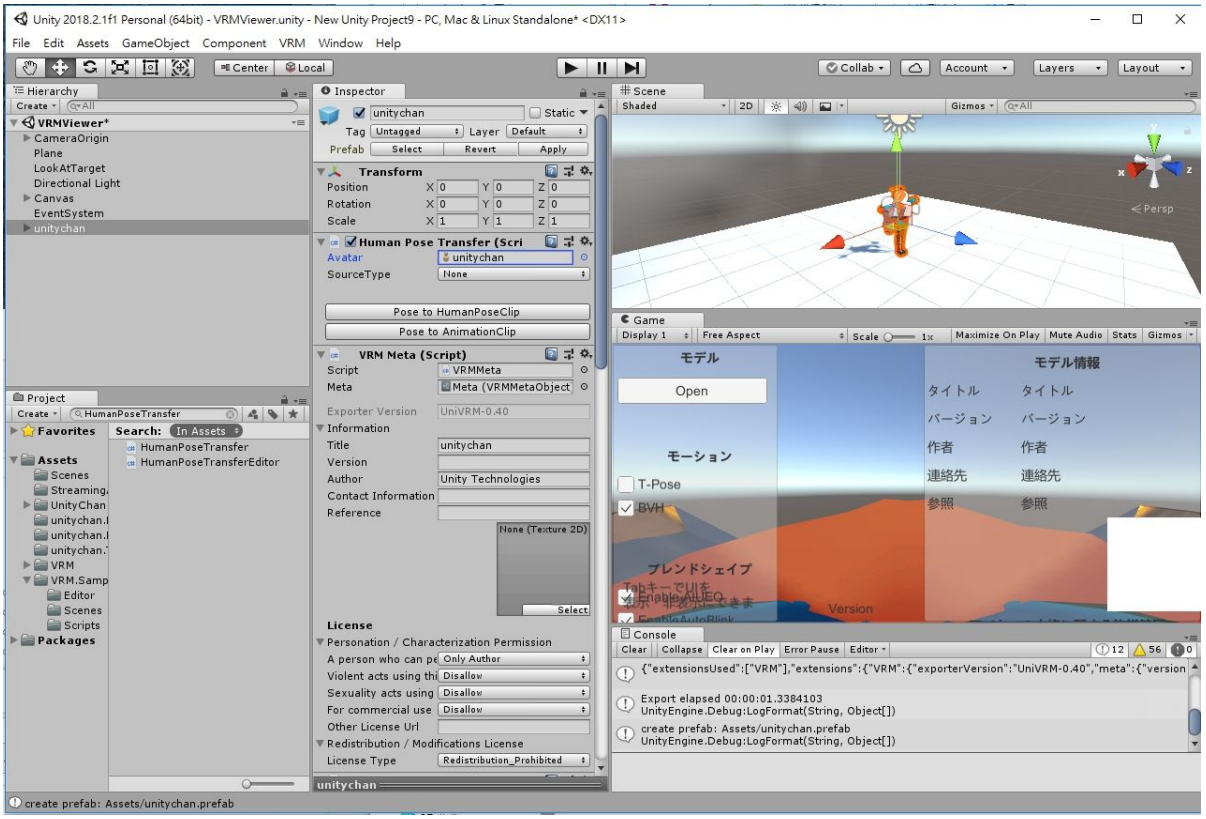
【c. Secondary设定，包含弹簧摇动骨架，碰撞】

VRM的这套开源Spring bones 系统也可以免费用于任何专案与模型，并不是只能用在VRM

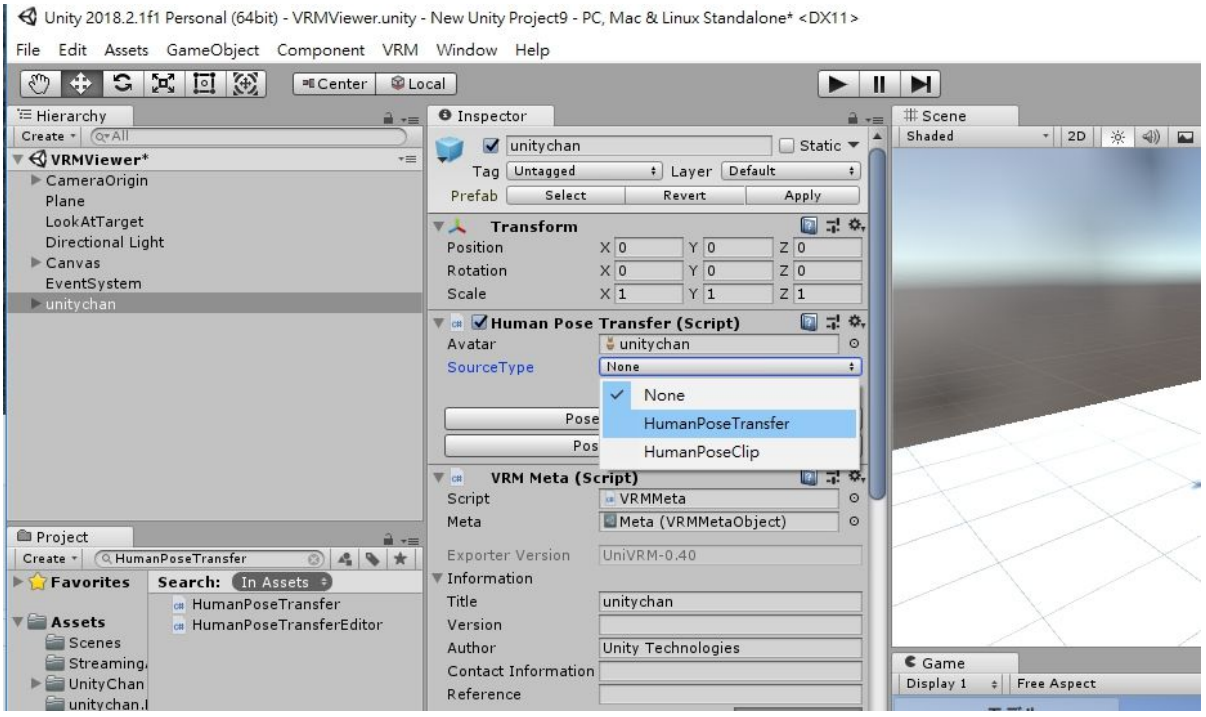
c-1.点选VRM官方的Example，VRMViewer，作为物理测试用，可以不必存档scene



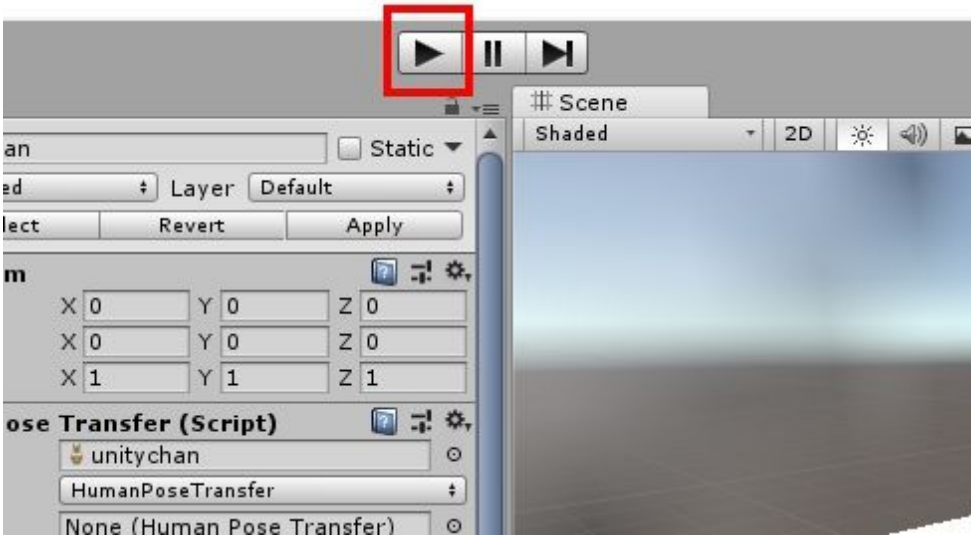
c-2.Secondary设定只能在gameObject
将~.prefab拖拉至Hierarchy视窗成为gameObject
在Project视窗搜寻出HumanPoseTransfer的script,
将HumanPoseTransfer拖拉进gameObject,
会出现与~.prefab中断连结的警告， 点选继续



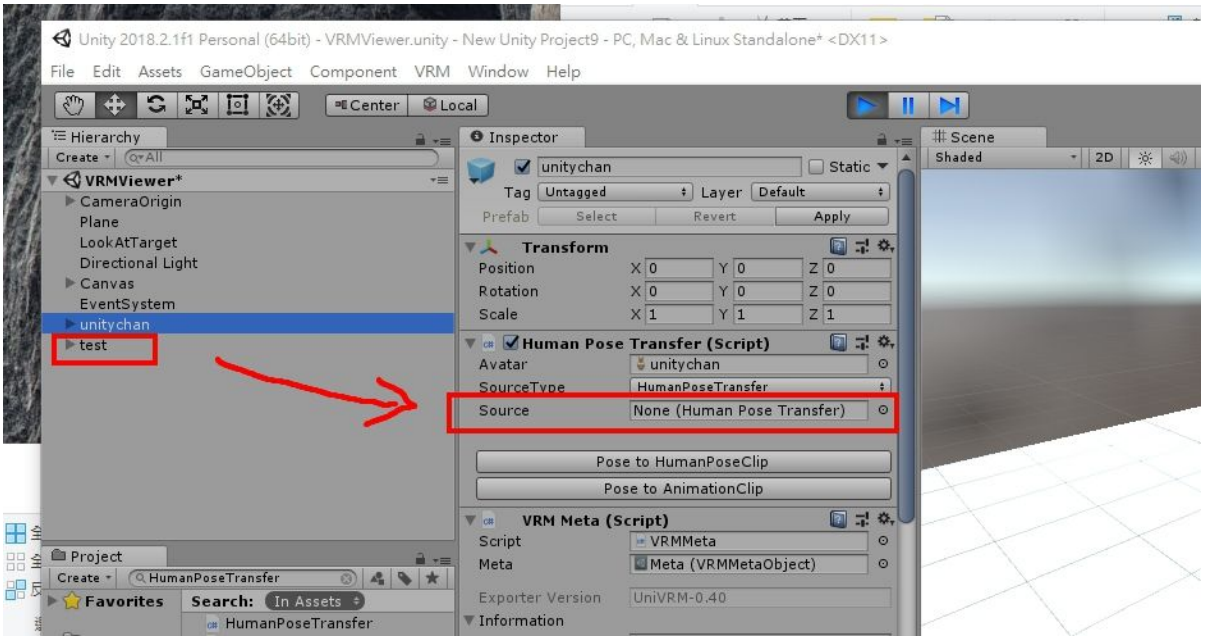
c-3.HumanPoseTransfer (Script)的SourceType选择HumanPoseTransfer



c-4点选play，注意所有在play进行中的操作，除了对Assets资料夹内的变动，其他变动都会在play结束后，回到play前的状态



c-5. Play之后会出现test gameObject，将之拖拉到unitychan的红框处，之后就可以看到UnityChan开始跳舞



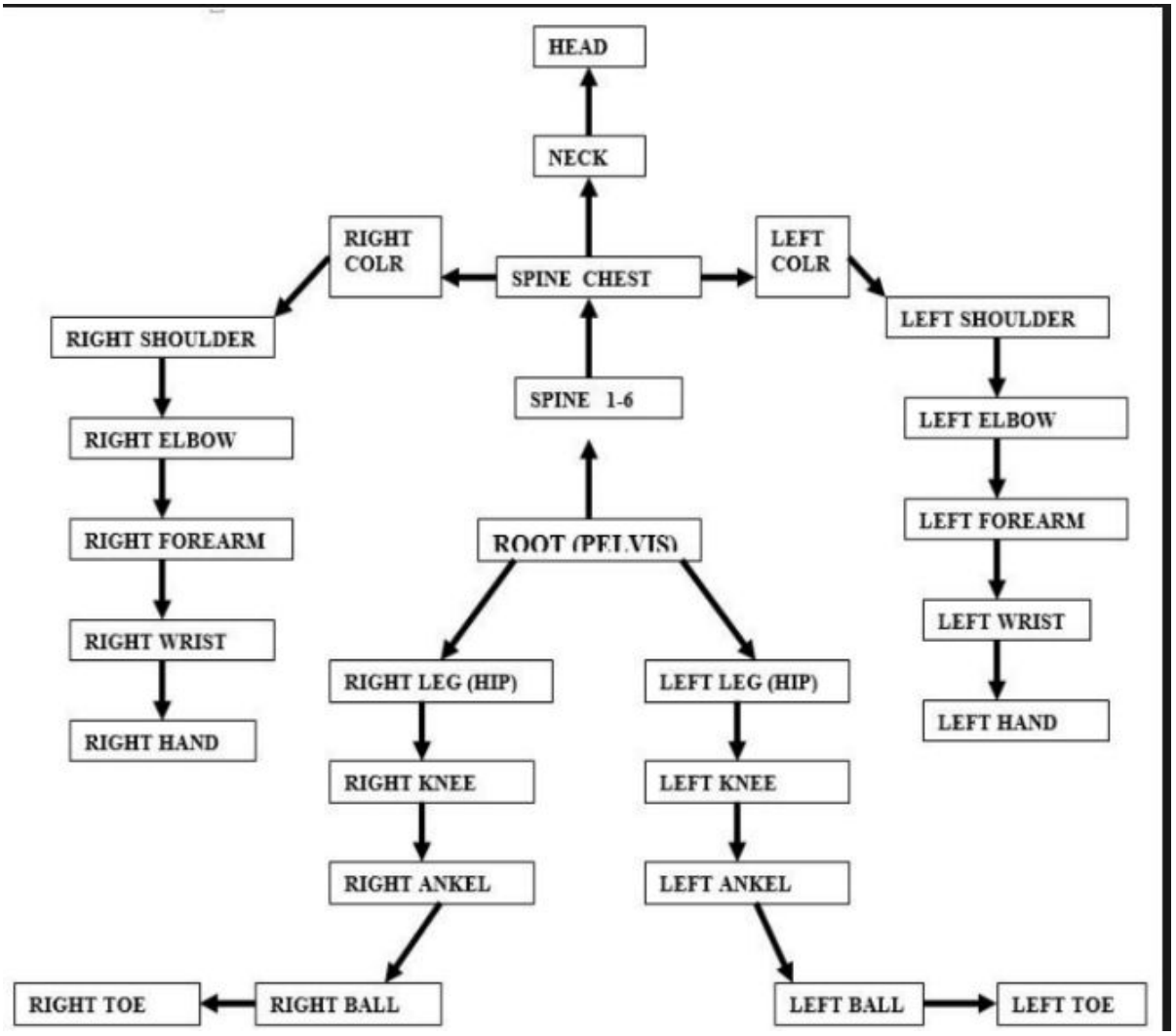
这时的Unity Chan会看到头发与衣服没有碰撞，也不会飘动，观察得到以下结论

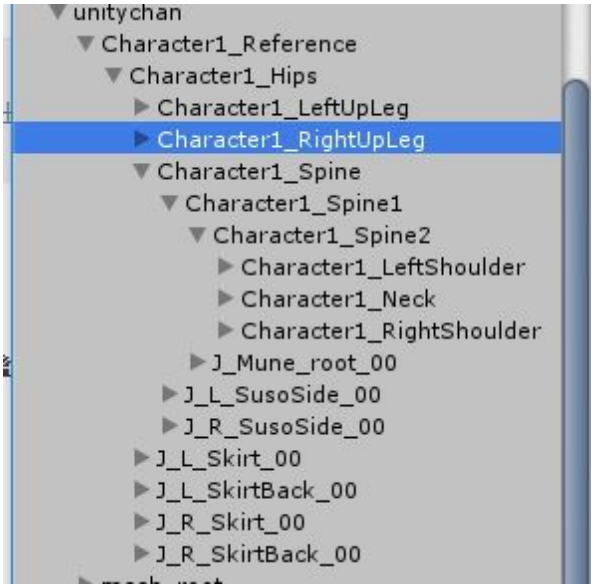
- 头发
  根部:头部
  明显不可互相穿过的对象: 头部，肩部，上臂，下臂，躯干
- 短裙前摆
  根部:腹部附近
  明显不可互相穿过的对象:大腿
- 短裙后摆
  根部:腹部附近
  明显不可互相穿过的对象:大腿

c-6.解除play状态，查看Unity Chan的gameObject结构

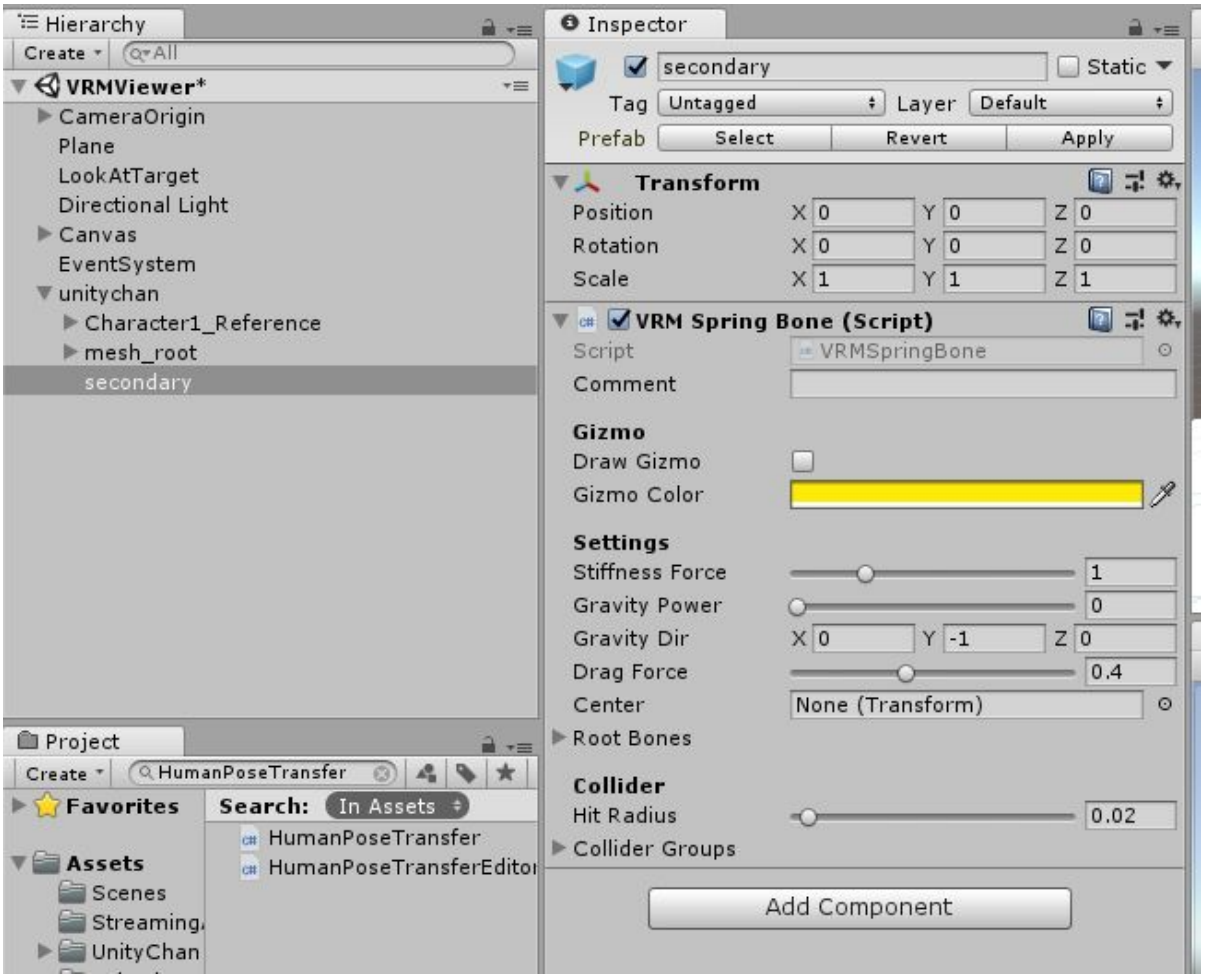
- Character1_Reference :骨骼所在

通常骨骼的结构与命名如图，树状阶层从root 或pelvis 或hips开始分成上下半身，各家模型略有不同，不过大致如此，这部分需要观察一下那些是骨骼，那些是头发衣服，通常命名都很清楚。

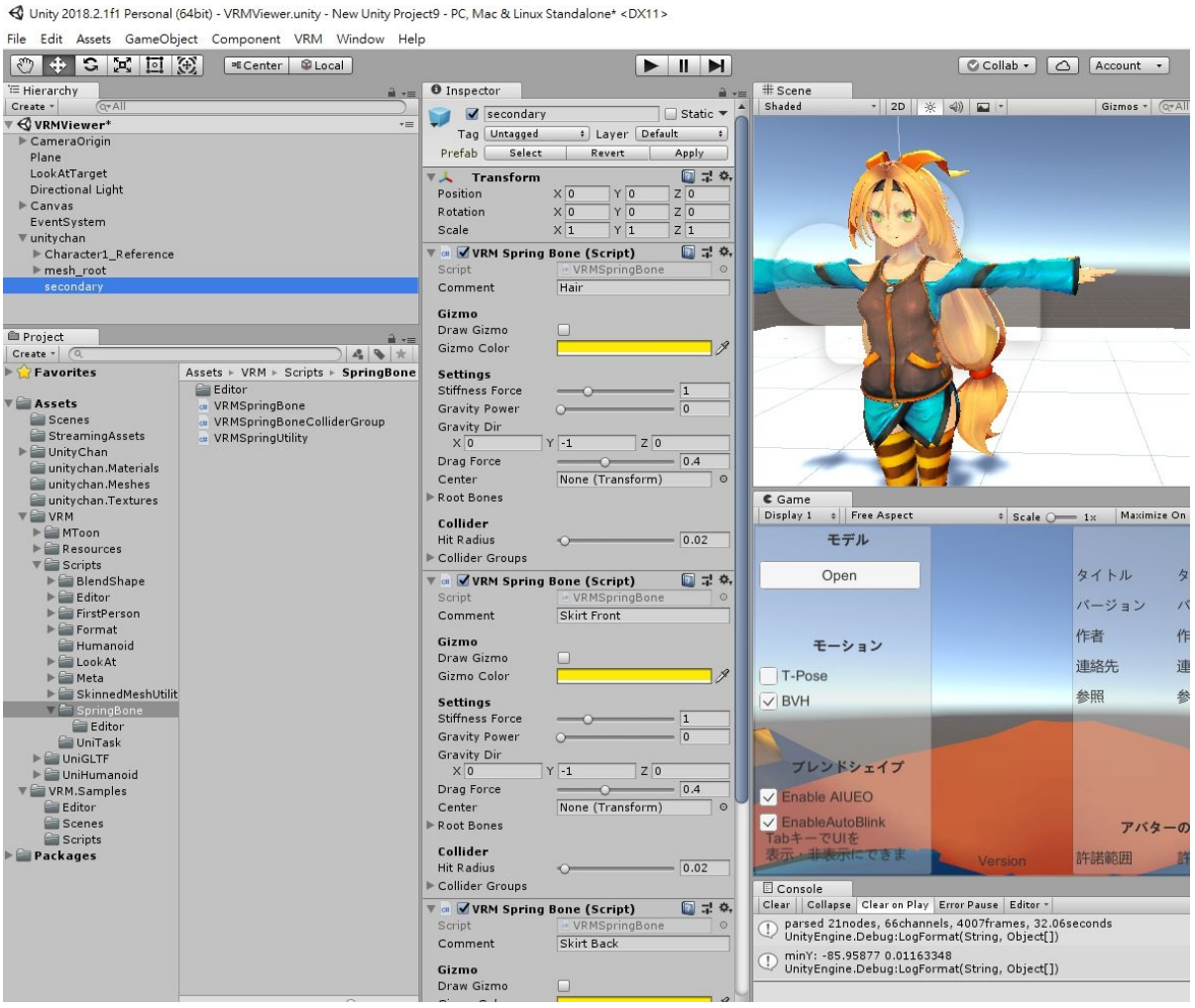




- mesh_root: Shaders的所在
- secondary: 物理与碰撞设定的所在，目前只有一个空壳



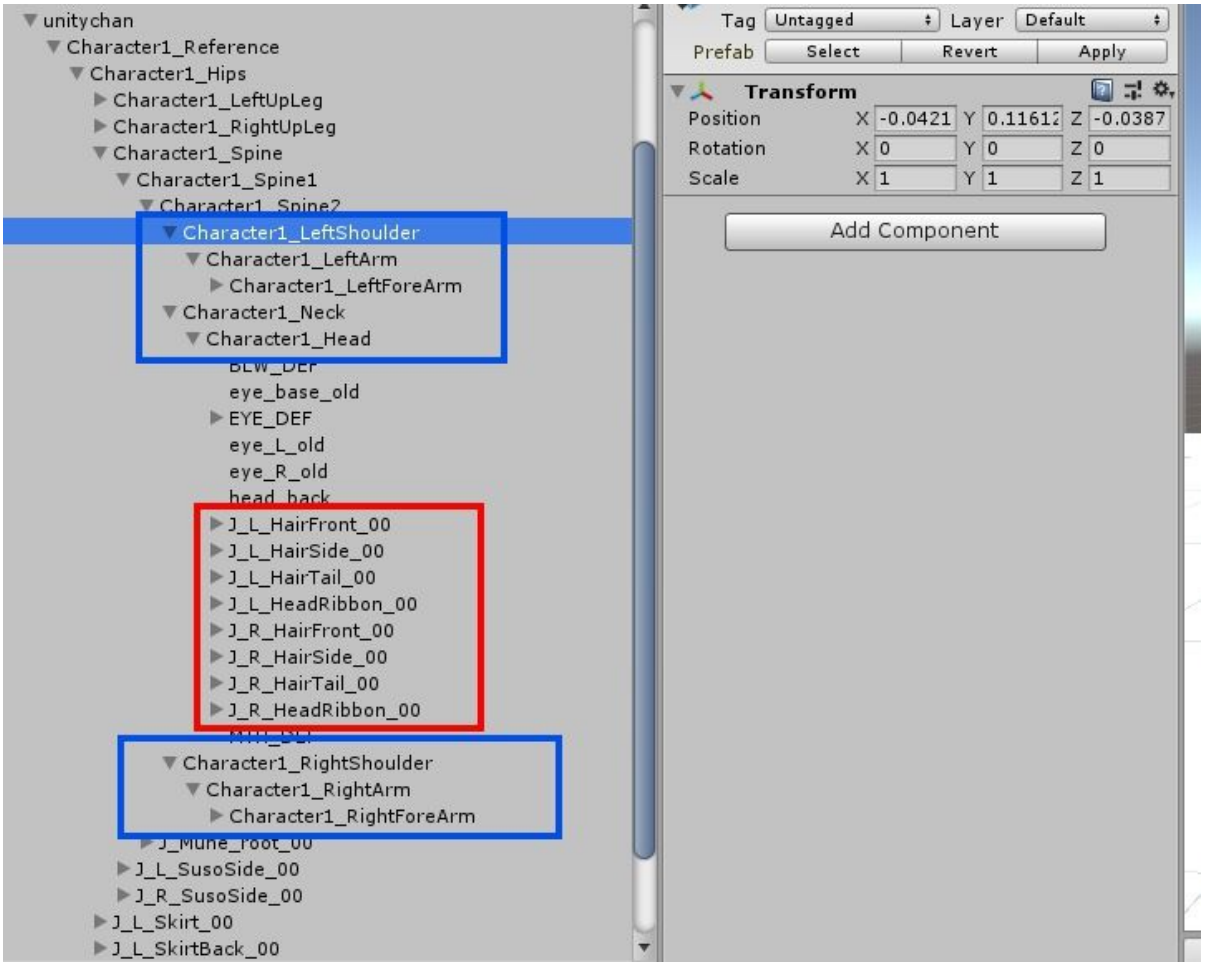
c-7 一个VRM Spring Bone是一群摇动与碰撞的区块，按照c-5的分析，这边会需要3个
找到VRM Spring Bone的script所在，直接拖拉进secondary建立Hair，Skirt Front，Skirt Back 3个



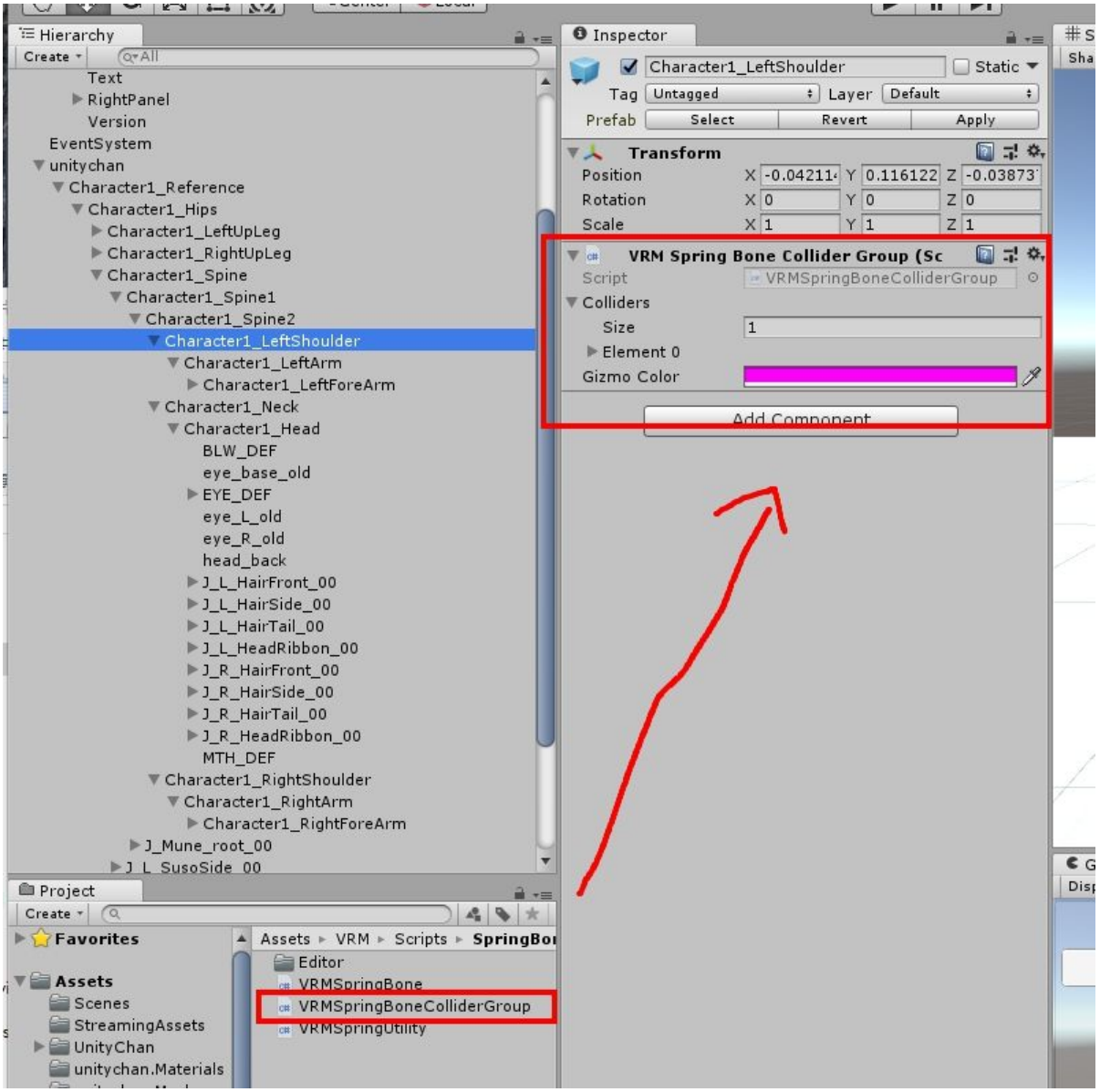
说明一下各参数定义:
Comment: 注解，可以像我这样取名
Draw Gizmo 勾选可以看到摆动参考球
Gizmo Color 摆动参考球的颜色
Siffness Force 柔软度，值越大越硬
Gravity Power 重力，值越大越重
Gravity Dir 重力方向 0,-1,0为向下
Drag Force 拉力，值越大越轻飘飘
Center 很抱歉，我也不知道做什么用的，似乎不常用
Root Bones 设定一群摆动对象的根部
Hit Radius 摆动对象的碰撞体积
Collider Groups 设定一群摆动对象，不再这里的就不会与摆动对象碰撞

c-8 这边只设定头发作为范例

红色为需要加入Root Bones 的摆动对象根部
蓝色为碰撞对象，视穿模情况可能需要加入更多，但这边只先加入这些

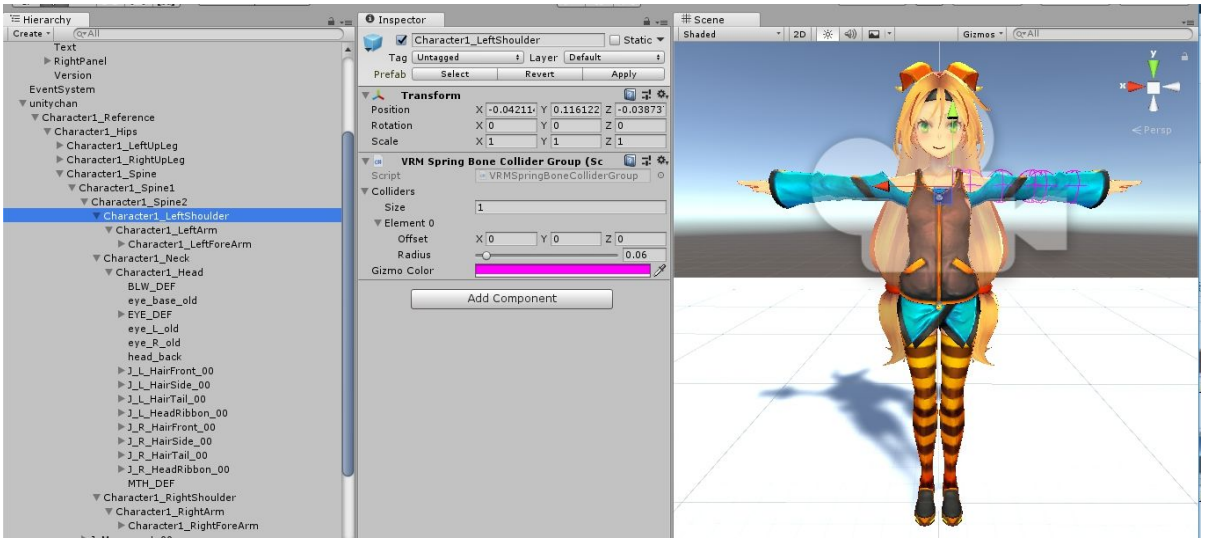


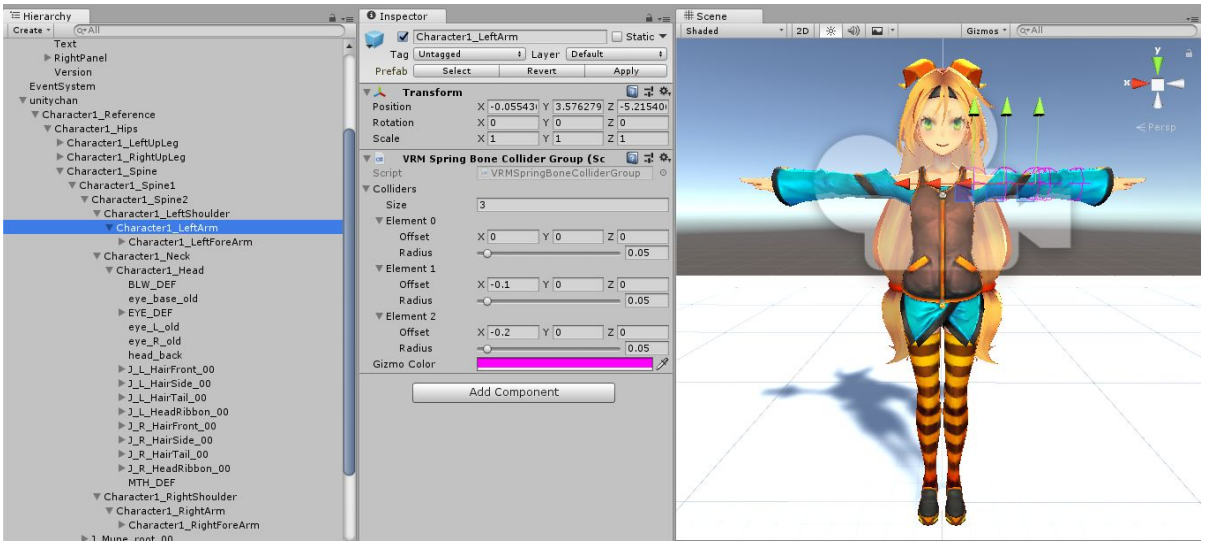
c-9. 将3个左手bones加入VRMSpringBoneColliderGroup这个Script

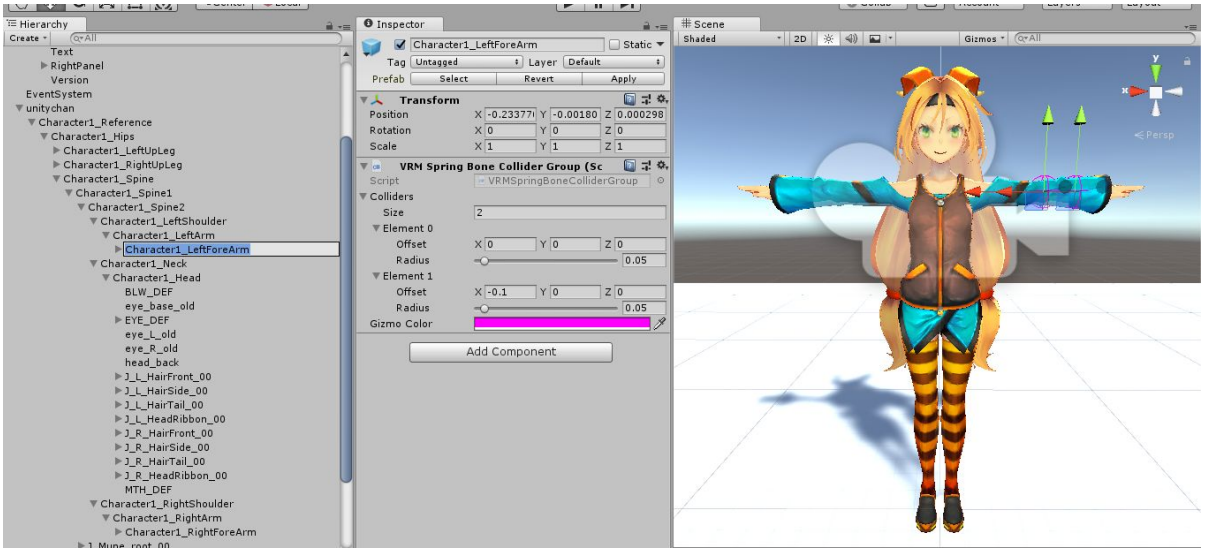


会看到这个碰撞体积过大，并且松散

调整大小，并增加碰撞球体

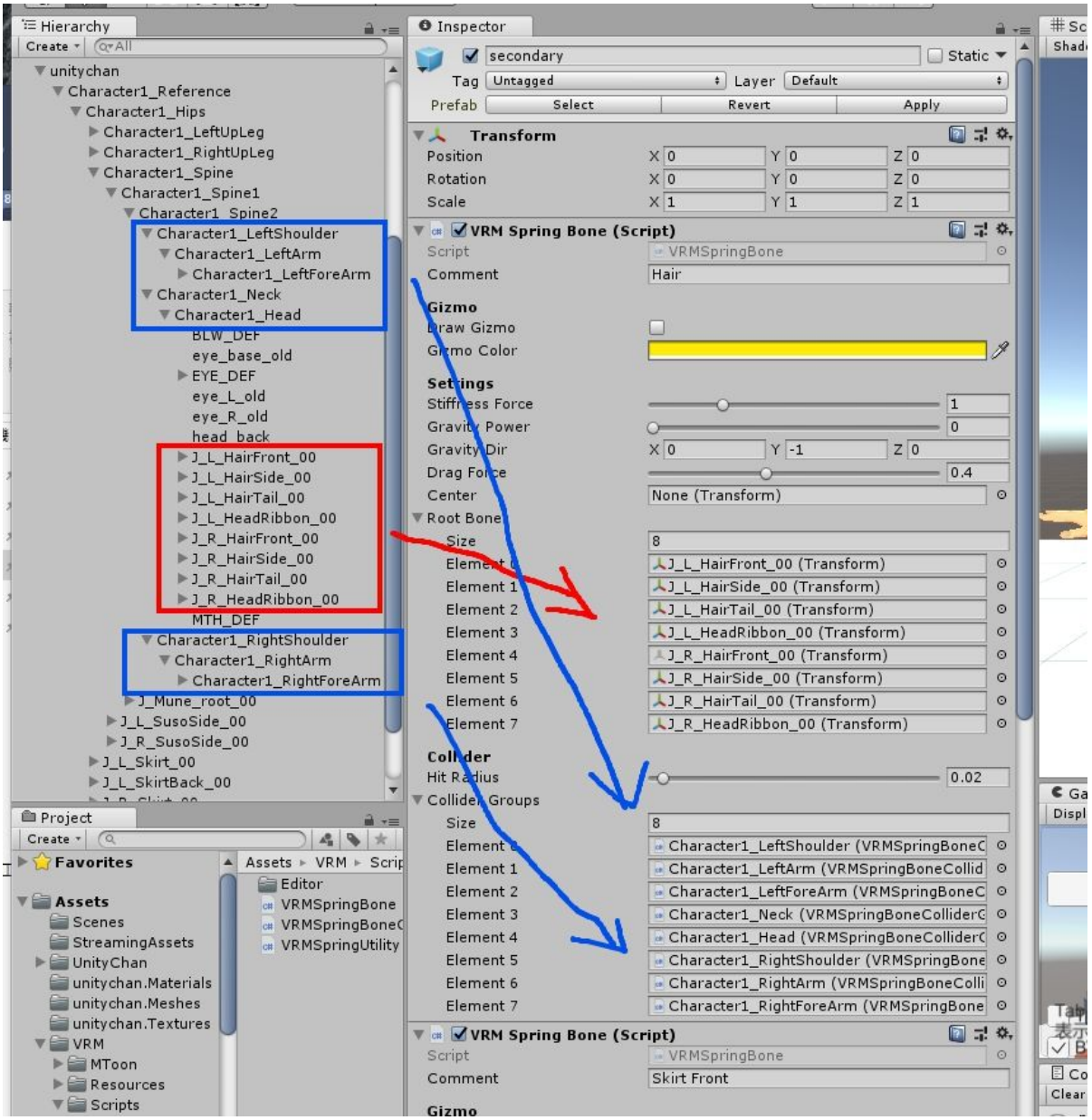




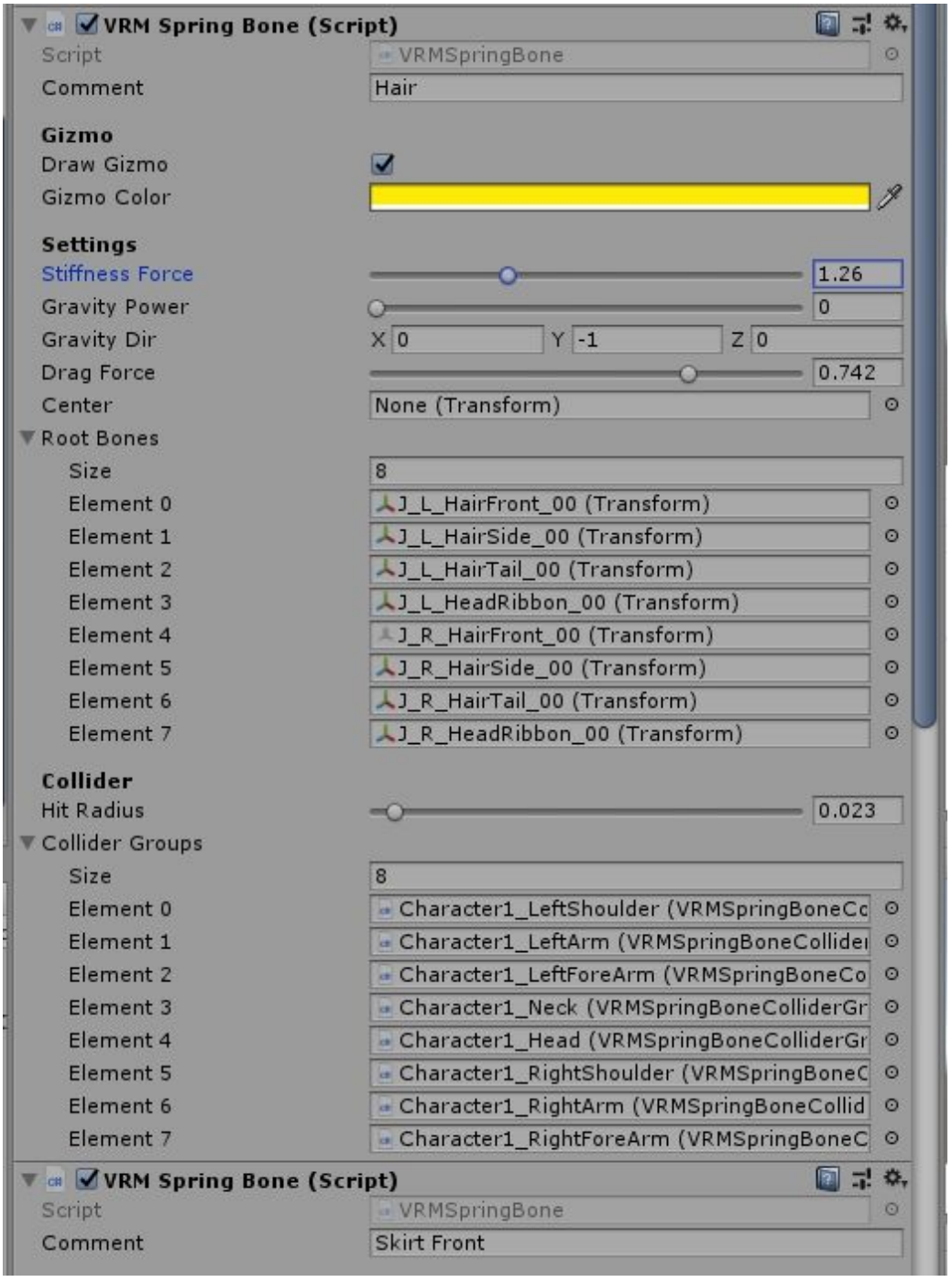


c-10. 将各处蓝色碰撞对象设定完成

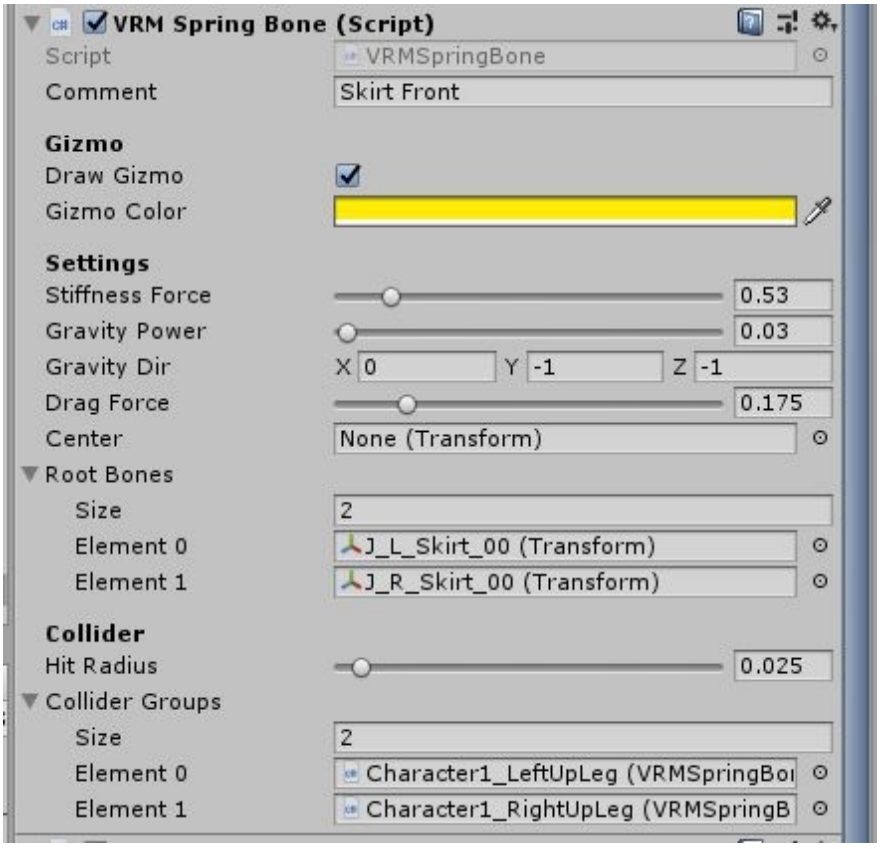
c-11. 如图设定好Root Bones与Collider Groups，其他数值可以重复c-4,c-5，在play中调整


Hierarchy
Create
unitychan
Character1_Reference
Character1_Hips
Character1_LeftUpLeg
Character1_RightUpLeg
Character1_Spine
Character1_Spine1
Character1_Spine2
Character1_LeftShoulder
Character1_LeftArm
Character1_LeftForeArm
Character1_Neck
Character1_Head
BLW_DEF
eye_base_old
EYE_DEF
eye_L_old
eye_R_old
head_back
J_L_HairFront_00
J_L_HairSide_00
J_L_HairTail_00
J_L_HeadRibbon_00
J_R_HairFront_00
J_R_HairSide_00
J_R_HairTail_00
J_R_HeadRibbon_00
MTH_DEF
Character1_RightShoulder
Character1_RightArm
Character1_RightForeArm
J_Mune_root_00
J_L_SusoSide_00
J_R_SusoSide_00
J_L_Skirt_00
J_L_SkirtBack_00
Inspector
secondary
Static
Tag
Untagged
Layer
Default
Prefab
Select
Revert
Apply
Transform
Position
Rotation
Scale
VRM Spring Bone (Script)
Script
VRMSpringBone
Comment
Hair
Gizmo
Draw Gizmo
Gizmo Color
Settings
Stiffness Force
Gravity Power
Gravity Dir
Drag Force
Center
None (Transform)
Root Bone
Size
Element 0
Element 1
Element 2
Element 3
Element 4
Element 5
Element 6
Element 7
J_L_HairFront_00 (Transform)
J_L_HairSide_00 (Transform)
J_L_HairTail_00 (Transform)
J_L_HeadRibbon_00 (Transform)
J_R_HairFront_00 (Transform)
J_R_HairSide_00 (Transform)
J_R_HairTail_00 (Transform)
J_R_HeadRibbon_00 (Transform)
Collider
Hit Radius
Collider Groups
Character1_LeftShoulder (VRMSpringBoneC
Character1_LeftArm (VRMSpringBoneCollid
Character1_LeftForeArm (VRMSpringBoneC
Character1_Neck (VRMSpringBoneColliderG
Character1_Head (VRMSpringBoneColliderC
Character1_RightShoulder (VRMSpringBone
Character1_RightArm (VRMSpringBoneColli
Character1_RightForeArm (VRMSpringBone
VRM Spring Bone (Script)
Script
VRMSpringBone
Comment
Skirt Front
Gizmo
Project
Create
Favorites
Assets
Scenes
StreamingAssets
UnityChan
unitychan.Materials
unitychan.Meshes
unitychan.Textures
VRM
MToon
Resources
Scripts
Assets ▸ VRM ▸ Scrip
Editor
VRMSpringBone
VRMSpringBoneC
VRMSpringUtility

最终数值

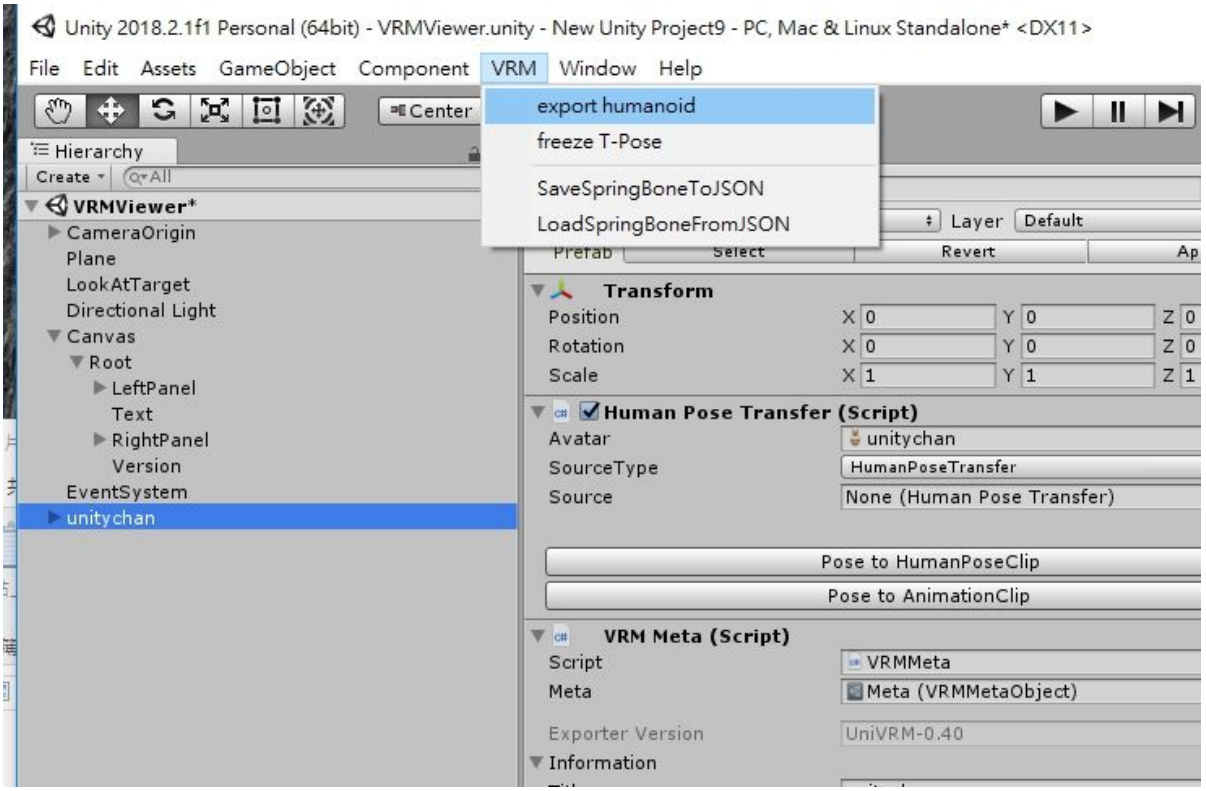

VRM Spring Bone (Script)
Script
VRMSpringBone
Comment
Hair
Gizmo
Draw Gizmo
Gizmo Color
Settings
Stiffness Force
1.26
Gravity Power
0
Gravity Dir
X 0
Y -1
Z 0
Drag Force
0.742
Center
None (Transform)
Root Bones
Size
8
Element 0
J_L_HairFront_00 (Transform)
Element 1
J_L_HairSide_00 (Transform)
Element 2
J_L_HairTail_00 (Transform)
Element 3
J_L_HeadRibbon_00 (Transform)
Element 4
J_R_HairFront_00 (Transform)
Element 5
J_R_HairSide_00 (Transform)
Element 6
J_R_HairTail_00 (Transform)
Element 7
J_R_HeadRibbon_00 (Transform)
Collider
Hit Radius
0.023
Collider Groups
Size
8
Element 0
Character1_LeftShoulder (VRMSpringBoneCc
Element 1
Character1_LeftArm (VRMSpringBoneCollide
Element 2
Character1_LeftForeArm (VRMSpringBoneCo
Element 3
Character1_Neck (VRMSpringBoneColliderGr
Element 4
Character1_Head (VRMSpringBoneColliderGr
Element 5
Character1_RightShoulder (VRMSpringBoneC
Element 6
Character1_RightArm (VRMSpringBoneCollid
Element 7
Character1_RightForeArm (VRMSpringBoneC
VRM Spring Bone (Script)
Script
VRMSpringBone
Comment
Skirt Front

前裙摆的设定



结果

c-12最后都完成后，记得输出VRM作为存档



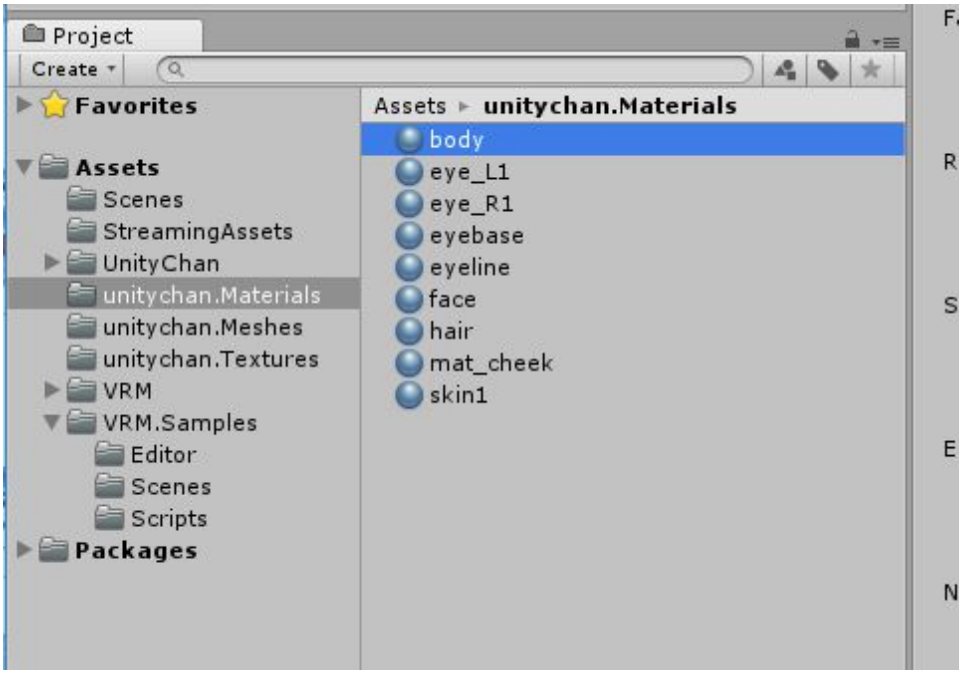
官方文档
https://dwango.github.io/en/vrm/univrm/components/univrm_secondary/

【d. Shaders修改】

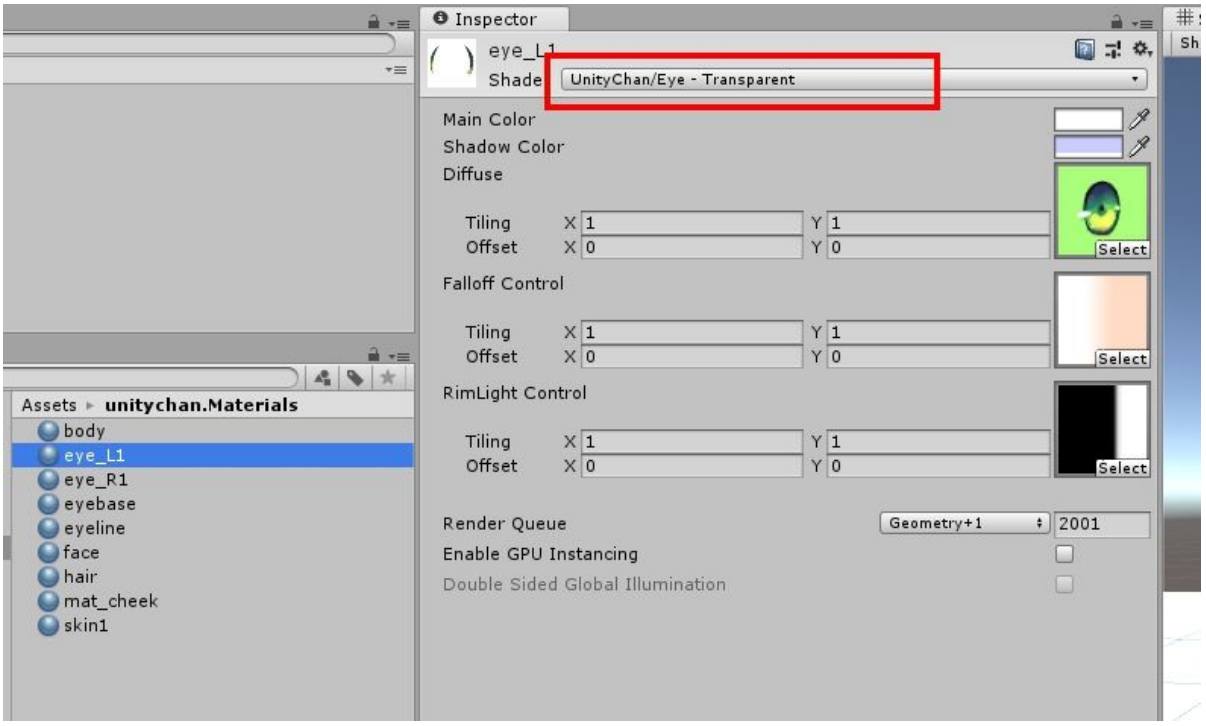
Shaders修改，对于新转换的FBX，这步非常重要，如果漏掉这步，直接使用你的VRM可能载入后就会看到这样，纹理，透明度都错误

因为VRM只会完整搭载vrm通用的Shaders，如果使用了其他不通用的Shaders，以Unity Chan来说，都是Unity Chan自己的Shader，其他软体没有，读取后就会变成这样。
在编辑器看不出异状，也是因为载入Unity Chan的 unitypackage中也载入了Unity Chan自己的Shader，因此可以正常读取。
如果希望能够个软体通用，就要设定成vrm通用的Shaders

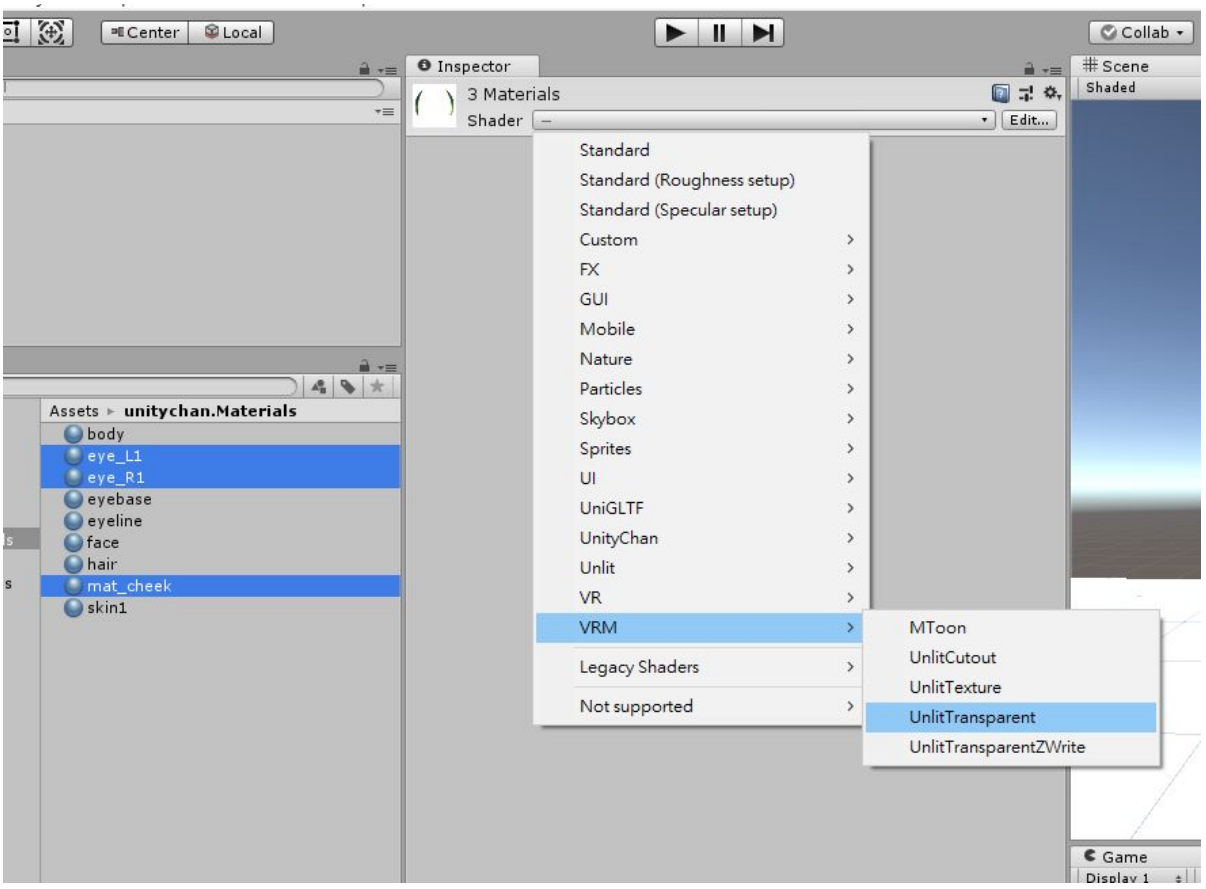
d-1. 找到读取~.vrm会自动产生的资料夹 ~.Materials



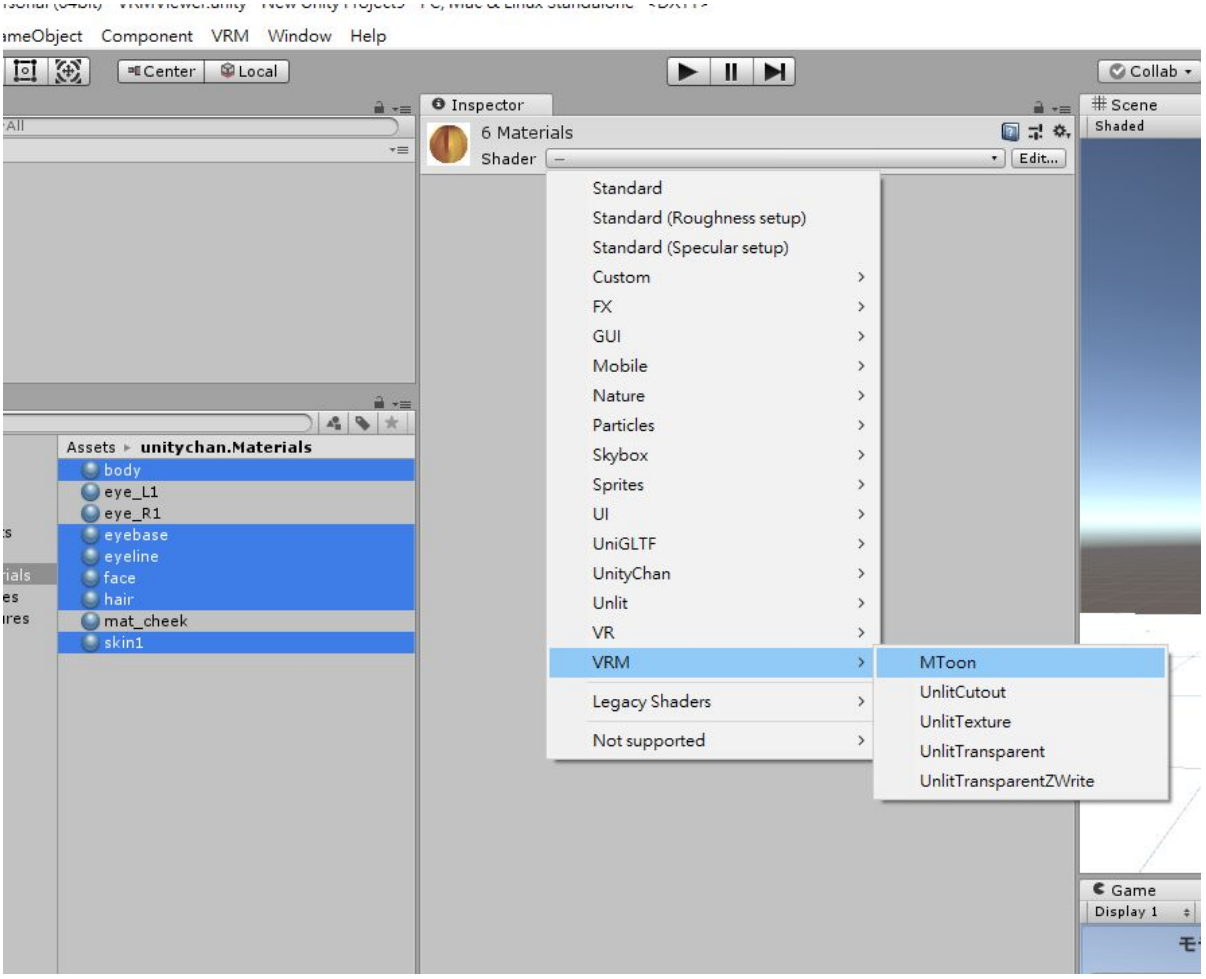
观察每一个后会发现，都是UnityChan自己的Shader，
其中eye_L1，eye_R1，mat_cheek是Transparent，也就有透明度的



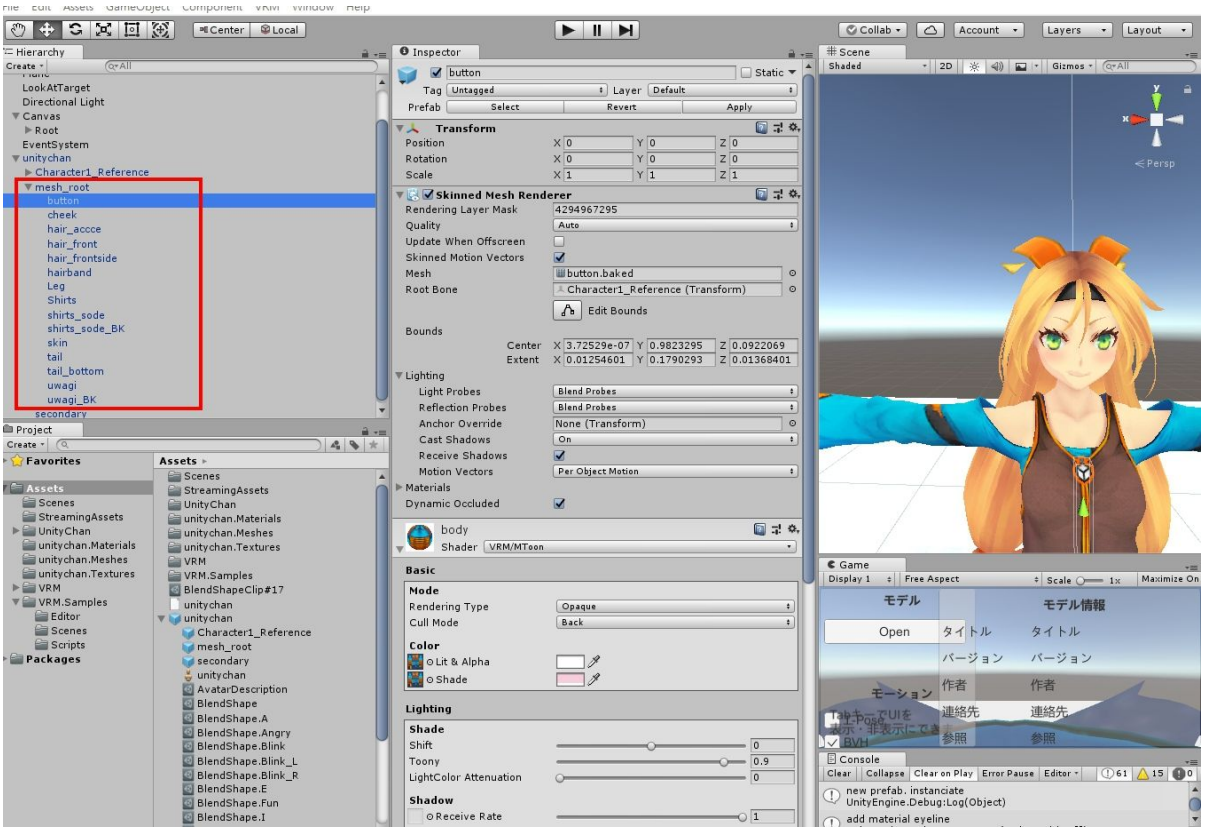
d-2. 将带有透明的换成VRM-->UnlitTransparent



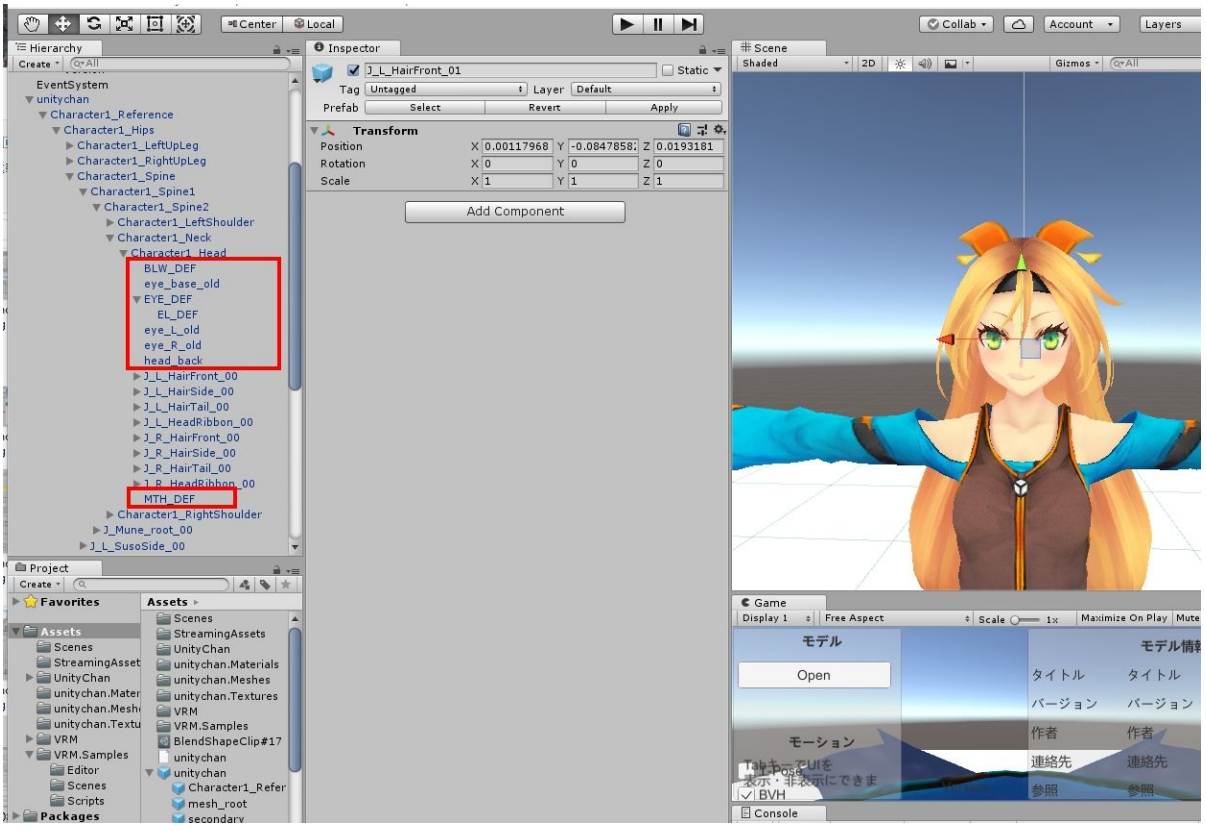
d-3. 将其他换成VRM-->MToon



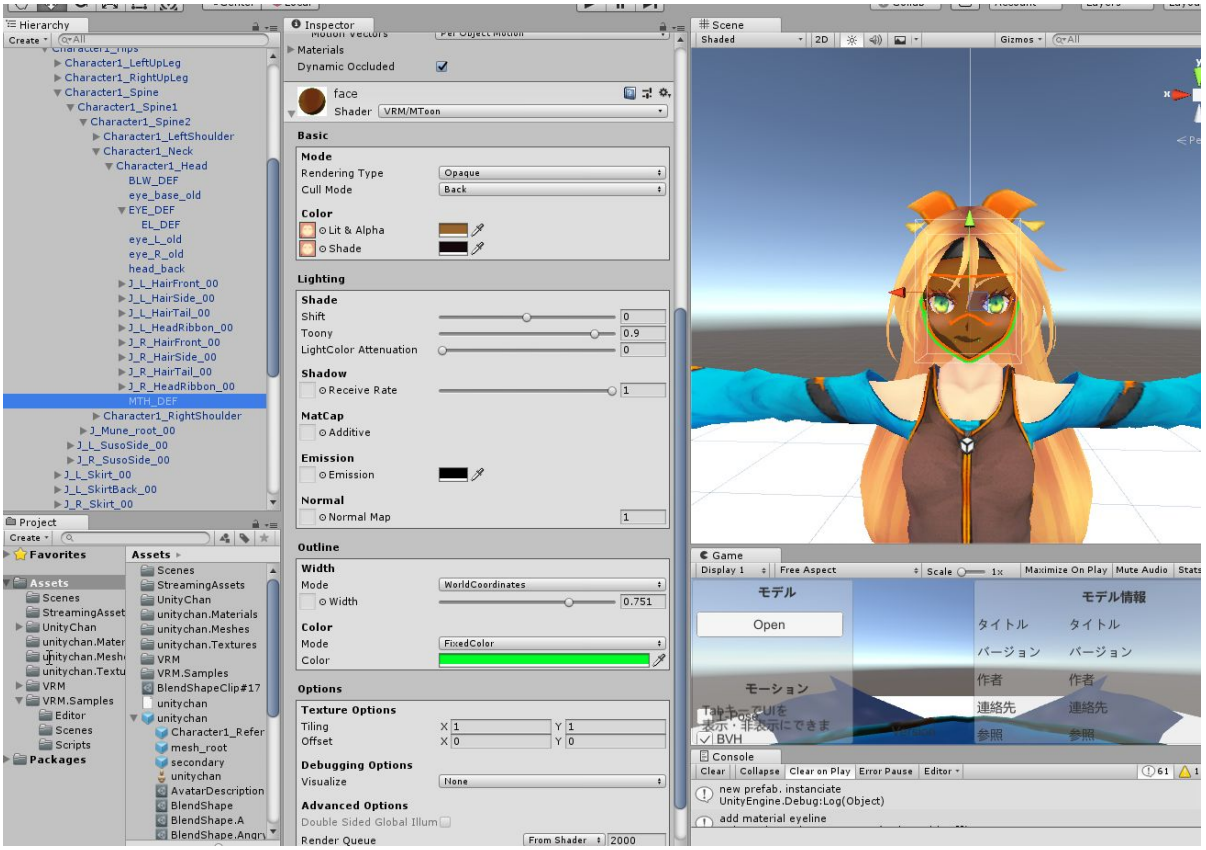
d-4. 将~.prefab拖拉至Hierarchy视窗成为gameObject
mesh_root内全是脸部以外各贴图的Shader


Hierarchy
Inspector
Scene
Game
Project
Console
button
Transform
Skinned Mesh Renderer
body
Shader
VRM/MToon
Basic
Shade
Lighting
Open
モデル
モデル情報
タイトル
バージョン
作者
連絡先
参照
モーション
BVH

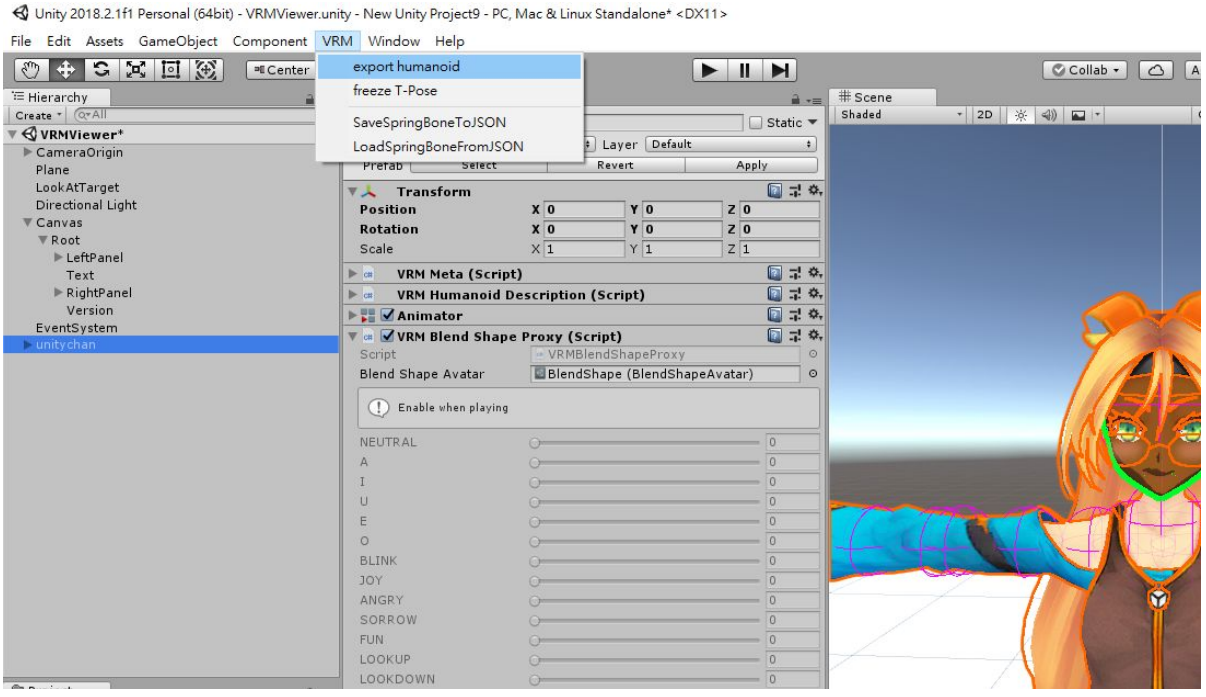
脸部则在Head内


Hierarchy
Inspector
Scene
Game
Project
Console
J_L_HairFront_01
Transform
Add Component
Character1_Head
BLW_DEF
eye_base_old
EYE_DEF
EL_DEF
eye_L_old
eye_R_old
head_back
MTH_DEF
Open
モデル
モデル情報
タイトル
バージョン
作者
連絡先
参照
モーション
BVH

d-5. 可以按需求加入描边，或是加深阴影，或是换色，如果做错了Ctrl+z可以回复上一动
或是在确定可以被软体支援的情况下，换成其他的Shader
我这边随便选个色作为示范



d-6. 最后一样记得输出成VRM存档



官方文档
https://dwango.github.io/en/vrm/univrm/shaders/univrm_shaders/